1 Web Services & SOA Fundamentals

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.

A web services is any piece of s/w that makes itself available on the internet & uses standardized XML messaging system.

eg: Client invokes a web service by sending an XML message then wait for corresponding XML responses. Because all communication is in XML.

Web services are not tied to any one operating system or programming lang.

Web Services are self contained, modular, distributed, dynamic application that can be ~~distributed~~ described, published, located or invoked other N/w to create process.

Web services are generally -

* Application to appl^n communication.
* Comm^n over network.
* Platform independent.
* Open protocol is used for establishing communication. Open protocol is publicaly available.
* Web services are self contained & self describing.
* Web services can be discovered using UDDI (Universal Description, Discovery & Integration)
* HTTP & XML is basis for web services.

Scanned by CamScanner

Web services has two types of uses.

① Reusable appl^n component
There are some application which need very often. So web services can offer application component like the weather report, language translation service.

② Connect existing S/w -
Web services can help to solve the interoperability problem by giving different applications a way to link their data. With web services you can exchange data between different appl^n & diff^n platforms.

Web services are self contained -
On the client side, no additional s/w is required. The XML & HTTP client support is enough to get you started.
On the server side, a web server, servlet engine are required. The client & server can be implemented in the diff^n environment.

Web services are self describing -
The client & server need to recognize only the format & content of request & response messages. No external meta-data repositories or code generation tools are required.

No external tools or repositories are required

Web services are platform independant - web services does not required specific programming lang. & operating system.

Client can across a varity of platforms & prog. lang.

SaaS (Software as a Service) -
SaaS is a way of delivering application over the internet as a service. Instead of installing & maintaining s/w, you simply access it via the internet, freeing yourself from complex s/w & H/w management.
Saas applications are sometimes called web-based s/w, on-demand s/w or hosted s/w. Saas appl^n runs on the Saas provider's server. The provider manages access to the appl^n, including security, availability & performance.
SaaS is closely related to the ASP (Application Service provider) & on-demand computing s/w delivery models.
Saas is becoming an increasingly prevalent delivery model as underlying technology that support web services & SOA (Service Oriented Architecture) & new developmental approaches such as Ajax.

Advantages of SaaS
① It is easy to implement.
② Easy to update & debug.
③ Less expensive - SaaS has different regarding costs. It usually resides in a shared or multitenal environment where the H/w & s/w license costs are low compared

Scanned by CamScanner

multitenant - multiple Client.

compared with traditional model.
Maintenance cost is also reduced.
Some SaaS has no license fees means lower initial cost.

Painless Upgrades - Because SaaS provider manages all updates & upgrades. The SaaS provider also manages availability. So there is no need for customers to add H/w, S/w

~~Seamless~~ Integration - (multiple Client)
SaaS vendors with true multitenant architecture can scale indefinitely to meet customer demand. There are many API that let you integrate with existing system to other system.

High Adoption -
SaaS applications are available from any computer or any device - any time, anywhere. Becoz most people are familier with using internet to find what they need.

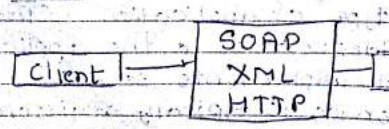
Difference between Web Services & Web Appl^n.
Web Appl^n - Any appl^n which resides on a server, use by human which uses web pages as presentation layer.
All user interactivity is done through web pages, but all data is stored & manipulated on the server.
It consist of GUI. The web appl^n as completed user interface.

* Web service are server based appl^n which may be accessed over the Network, but it is used for interation with other program.
Thus it will have a clearly - defined API which consist of providing responses to HTTP GET & POST request.
The web services not necessarily have a GUI.
In web service, it receives all results of GET & POST request as a string of XML.
The web services more intended to be appl^n component or "libraries" which can be used by the other application.

Characteristics Of Web Services -
① A web service is a web resource. You access web service using platform independent & language independent, open protocol such as HTTP.
② A web service provider provides an interface - a web API that can be called from another program. This application to appl^n programming interface can be invoked from any type of application. The web API provides access to the application logic that implements service.
③ A web service is typically registered & can be located through a web service registry. A registry enables service consumers to find services that match their needs. These service consumer may be human or other application.

Scanned by CamScanner

(4) Web services supports loosely coupled connections between system.
Means, A consumer of web service is not tied to web service directly. The web service interface can change over time without compromising the client's ability to interact with the service. In tightly coupled system implie that the client & server logic are closely ~~updated~~ tied to one another, implying that if one interface changes, the other must also be updated.
Due to loosely coupled system, the S/W makes more ~~map~~ manageable.
(4) Appl^n to Appl^n communication over N/W.
(5) Self contained & self describing.
(6) Do not provide users with GUI.
(7) Share business logic, data & processing through a programmatic interface across a network.
(8) Web service uses XML, SOAP, WSDL, UDDI.



Simple object access protocol
WSDL - web service description lang
UDDI - Universal Description, discovery & Integration.

Web Service Interface -
With web services interface (WSI) S/W application can exchange data over the N/W by using standard like XML & HTTP & thus provide interoperability & extensibility.
The extra information needed where the service is located (IP Address) & the msg format used by the service. Here is what a service description should contain -
(1) Interface Name
(2) Operation's name
(3) Operation I/P parameter
(4) Operation Return Value.
(5) Service msg format
(6) Service location (IP Address / URL).

The web service interface description looks depend on the msg format used by web service. The currently SOAP web services has a standardized interface description - WSDL (Web Servic Description Language).

Implementation is the carrying out, executing or practice of a plan, a method or any design for doing something. Imple^n is the action that must follow any preliminary thinking in order for something to actually happen.
Implementing web services appl^n is an easy way to integrate appl^n system together. Services can be implemented either in .NET / J2EE

Scanned by CamScanner

Ex - your customer information database is a standalone applⁿ, but you want your accounting applⁿ to be able to acce access to the customer data. You can create a web service for the customer database & then enable the accounting applⁿ as a web service, these two applⁿ can share information in an efficient manner.

Because web services are easily applied to existing applⁿ, new solution can be deployed quickly.

* SOA - (Service Oriented Architecture) :-

The SOA is essentially a collection of services. These services communicate with each other.

The SOA is a s/w design & s/w architecture design pattern based on distinct pieces of s/w providing applⁿ functionality as a service to other applications.

SOA have following key characteristics

① SOA services have self describing interfaces in platform independent. WSDL is the standard used to describe the services.

② SOA services communicate with message formally defined via XML Schema (also called XSD).

③ No. of application can look up the services in the registry & invoke the services. UDDI is the standard used for service registry.

④ Each services (SOA) has a quality of services (QoS) associated with it. Some of the key QoS elements are security requierements, such as - Authentication & authorization.

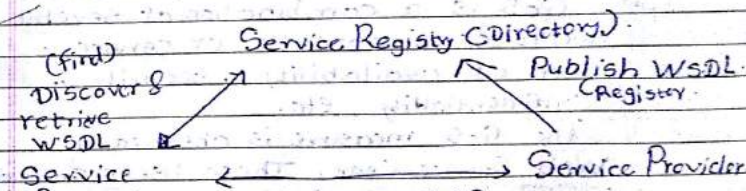
⑤ Loose Coupling. ⑥ Reuse of existing technology.

SOA is an archetecture. Provides conceptual design pattern for service based distributed system.

- manages many loosely coupled services developed using different technologies.

⑥ SOA supports composition (assembling) of services, reusability.

SOA implementation (Architecture).



Generally, the basic SOA describes the relationship between three kinds of participants The service provider, registry, service requestors. -

Scanned by CamScanner

* Service is a well defined fn that does not depend on the state of other services

Service provider publish their interface in the service registry. The client can use the service registry to find out about published interface & how to bind to the selected service.

Firstly, the service provider registers with the directory service, then client can query the directory service for providers & browse the exposed service capabilities.

Typically a directory service supports
1. a look-up service for client
2. Scalability of service model.
3. Dynamic composition of the services: The client can decide at runtime which service to use.

Quality Of Service (QoS)

QoS is a combination of several qualities or properties of service. Such as availability, security, confidentiality etc.

The QoS measure is observed by — Web Service User. These users are not human beings but programs that send request for services to web service provider.

Web Service QoS Requirements

The major requirements for supporting QoS in web services as follows:

1. Availability - Availability is the quality aspect of whether the web service is present or ready for immediate use. Availability represents the probability that service is available.
   Availability is associated with TTR (Time - To - Repair). TTR means represent the time to takes to repair a services that has failed.
   Larger value represent that the service is always ready to use while smaller value represent unpredictability of whether the service will be available at a particular time.
2. Accessability - Accessability is the quality aspect of a service that represents the degree it is capable of serving a web service request. There could be situation when a web service is available but not accessable.
3. Integrity - Integrity is the quality aspect of how the web service maintains the correctness of the interaction in respect to the source. Proper execution of web service transaction will provide the correctness of interaction. A transaction refers to a sequence of activities to be treated as a single unit of work. All the activity have to be completed to make the trantransaction successful.

Scanned by CamScanner

① Performance - The performance is the quality aspect of web service which measured in terms of throughput & latency. Higher throughput & lower latency values represent good performance of a web service.

Throughput represents the no. of web service requests served at a given time period.

Latency is the round-trip time between sending a request & receiving the respons

⑤ Reliability - Reliability the quality aspect of web service that represent the degree of being capable of maintaining the service & the service quality.

The no. of failures per month or year represents a measure of reliability of a web service.

Reliability refers to the assured & ordered delivery for messages being sent or received by service Requestor & service provider.

* Web Service Interoperability :-
Being able to accomplish end-user appl^n using different types of computer system, operating system & appl^n s/w interconnected by different types of local & wide area N/w.

The web services ~~gives~~ interoperability between services that are running on different platform & implemented using different prog. language.

A web service created in a .NET platform can be invoked by any client appl^n implemented in different programming language like Java.

Scanned by CamScanner

* RESTful Services :-

REST stands for Representational State Transfer. REST is based on principals, which are used in the largest distributed appl^n - the word wide web. The REpresentational State transfer architecture is an architecture that describe how the web should work. REST is neither product nor a standard.

REST Service is -

- ✓ Platform independent.
- ✓ Language independent.
- ✓ Standard based (HTTP).

Data & Functionality considered as resource

RESTful web services are built to work best on the web. In the REST architectural style, data & functionality are considered resources & are accessed using URI (Uniform Resource Identifier) typically link on the web. The resources are acted upon by ~~using~~ using a set of simple, well defined operation. REST architectural style is an client - server architecture & designed to use a stateless communication protocol typically HTTP. In REST architecture style, client & servers exchange representations of resources by using a standardized interface & protocol.

The following principals encourges RESTful applications to be simple, lightweight & fast -

① Resource Identification through URI -
A RESTful web services exposes a set of resources that identify the target of the interaction with its client. Resource are identified by URIs, which provide a global addressing space for resources & service discovery.

② Uniform Interface - Resource are manipulated using a fixed set of four - create, read, update, delete operations - PUT, GET, POST DELETE.
- PUT create new resource, which can be then deleted by using DELETE.
- GET - retrieves the current state of a resource.
- POST - transfer a new state onto a resource.

③ Self Descriptive messages -
Resources are decoupled from their representation so that their content can be accessed in a variety of formats - such as HTML, XML, plain text, PDF, JPEG etc. Metadata about resource is available & used.

④ Stateful Interaction through Hyperlink -
Every interaction with a resource is stateless i.e request msg is self contained. Stateful interaction are based on the concept of explicit state transfer. State can be embedded in response msg to point to valid future state of the interaction.

Scanned by CamScanner

Several technique exist to exchange state - such as URI Rewritting, Cookis & hidden form fields.

* Web Services Versus Components -
A component is a sw object, meant to interact with other component
- has set of functionalities. A component has a clearly defined interface.

Advantages - Of Web Services.

1. Interoperability - This the most important benifits of web service. web services are based on standard-based communication web services are vert platform independent.
2. Usability - Web services allow the buiseness logic of many different system to be exposed over the web. You can use web service what u want from the service registry.
3. Reusability - Web services are easy to reuse web service component.
4. Deployability - web services are deployed over standard internet technology. This makes it possible to deploy webservie even over the fire wall to servers running on the internet on the other side.
5. Reduce cost - Using web services to verify information instantly will save you time & money.
6. Improved efficiency by reducing time to integrate appl^n.
7. The ability to develop new appl's much faster than before.

* Shortcomings of Web Services:-
1. Security - Web services are exposed to the public using Http-based protocol. As web services publicaly available, it must be implemented using authentication & authorization mechanism & & using SSL-enabling encryption of the message

Scanned by CamScanner

for securing the usage. Adopting open security standards like XML encryption, XML signature etc.

② Although HTTP is simple protocol, they were not really meant for long term session. Typically, a browser makes an HTTP connn, requests a web page & then disconnect.
But to stay connected for an extended period of time, this is difficult for web service.

Assignment Test-1

① Define following terms - ① web Service ② SaaS

② Explain advan Software as service What are the advantages of SaaS.

③ Explain characteristics of web services.

④ Explain SOA. What are the ~~ch~~ key characteristics of SOA.

⑤ Explain advantages of web services, Disadvantages.

⑥ Explain REST full services with the principals.

⑦ Define QoS. Explain QoS Requirements.

(20)

Web Service Architecture:

The web service Arch. is based upon the interaction between 3 roles — Service provider, Service registry & service requestor. The interaction involves the publish find & bind operation.

The service provider define the service description for the web service & publishes it to a service registry or service requestor. The service req. uses a find operation to retrieve the service description locally or from the service registry & uses the service description to bind with service provider & invoke or interact with the web service implementation.

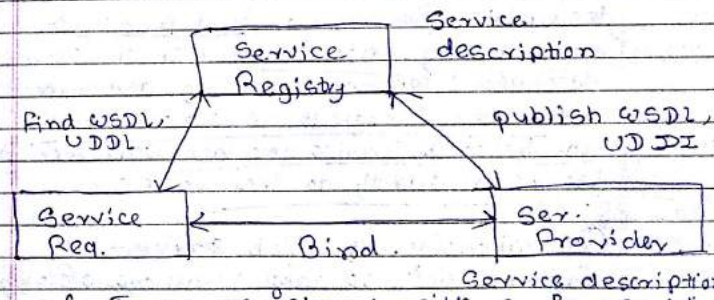


Fig- web service Role & operation

Roles in Web Service Architecture.

① Service Provider - This is the owner of the service. This is platform that hosts access to the service.

Scanned by CamScanner

Service prov. publish their service /interface in the service registry. First the service provider register with directory service.

② Service Requestor - This is the ~~fun^n bus~~ business that requires certain fun^n to be satisfied. This is the appl^n that is looking for & invoking or initiating an interaction with a service. The service req. role can be played by a browser driven by a person or a program without a user interface.

③ Service Registry - This is the searchable registry of service descriptions where the Service provider publish their service description. Service req. find services & obtain binding information (in the service description) for service during development.

The service reg. is an optional role in arch. becoz a service provider can send the description directly to the ser. req.

* Operations in a Web service -

For an appl^n to take advantage of web services, three behaviour must take place - publication of ser. desc., lookup or finding of serv. desc. & binding or invoking of serv. based on ser. desc.

① Publish - To be accessible, SD need to be published so that the SR can find it. Where it is published can vary depending upon the requirement of appl^n.

② Find - In the find op, SR retrieves a SD directly or queries the the service registry for a type of service required.

The find operation can be involved in 2 diff. lifecycle phase for ser. req. - At design time to retrieve the service's interface desc. for prog. development & at runtime to retrive the service binding & location descr. for invoking.

③ Bind - Serv. need to be invoked. In the bind op., the serv. re. invoke/initiate an interaction with the service at runtime using the binding details in SD to locate, contact & invoke service.

Core building blocks of web services.

Some of the main building block of web services are -

① SOAP - ~~SOPA~~ SOAP stand for Simple Object Access Protocol. SOAP is an xml based protocol for accessing web services.

SOAP lets one application invoke a RPC (Remote Procedure call) on another application or pass an object to a remote location using an XML message & the internet. Becoz xml is language & platform independent.

This is one of the most importaimportant protocol ususes in web services.

The Service Requestor send a SOAP

Scanned by CamScanner

message request to the web services & web service replies a SOAP message in respond. So both uses the SOAP message for communication.

② WSDL - Web Service description Language is an XML based language ~~language~~ to define web service & describe how to access them. Being XML-based WSDL is both machine & human readable. WSDL is used for describing the functionality offered by a web service. WSDL is nothing but XML document. WSDL is also used to locate web services. WSDL is integral part of UDDI. WSDL is developed by jointly by - Microsoft & IBM.

③ UDDI - Universal Description, Discovery & Integration.

As WSDL allows services to advertise themselves individually, UDDI provides a "directory service" for services & other information available on the web. UDDI is an XML-based standard for describing, publishing & finding web services. UDDI is a specification for a distributed registry of web services. The

* Standards & technologies available for implementing web services.

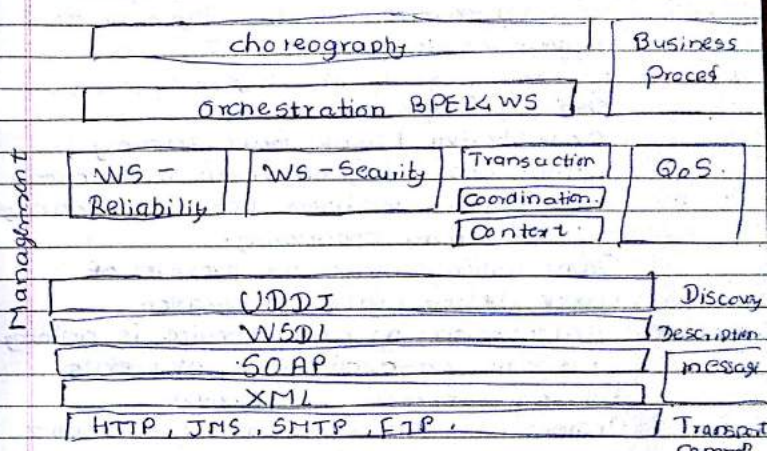


Fig - Web Service technology stack.

① Service Transport - The service transport layer delivers messages between applications. This layer usually implements HTTP, SMTP, FTP & newer protocol such as - Block Extensible Exchange Protocol (BEEP)

② XML messaging - This layer is responsible for encoding messages in a common XML format. Currently, this layer includes - XML-RPC & SOAP.

③ SOAP -

④ WSDL -

Scanned by CamScanner

BPEL - Business Process Execution Langu

Service Discovery - This service discovery layer registers services into a common repository & provides an easy publish / find mechanism. This layer is often implemented via UDDI

Relates to execution of specific business process

* Service Orchestration.
orchestration handles four primary requirements - asynchronous interaction, flow control, business transaction management & activity monitoring.
Orchestration carries the promise of cheaper, better, faster integration.

* orchestration in web service is nothing but the co-ordination of events between different web service.

* Orchestration directs & manages the on-demand services, to create a composite application or business process.

* Web service orchestration relates to execution of specific business process. WS-BPEL is language for defining process that can be executed on an orchestration engine. WS-BPEL is an xml based langu enabling users to describe business process activities as web service & define how they can be connected to accomplish specific task. WS-BPEL uses web service standard to describe business process activities.

* Choreography -
Rules that govern behavioral characteristics relating to how a group of web service services interact can be applied as a choreography. These rules include the sequence in which web services can be involved, conditions that apply to this sequence being carried out.
Web service choreography relates to describing externally observable interaction between web services. The
For web service choreography - WS-CDL is used. (Web Service Choreography Description Language).
WS-Choreography define the XML based business process modeling language that describe collaboration protocols of co-operating web service participants, in which services act as peers & interactions may be long lived & stateful

* WS-Reliability - Web Service Reliability is a SOAP based protocol for exchanging SOAP messages with guaranteed delivery, no duplicates & guaranteed msg ordering.
It defines reliable message delivery as the ability to guarantee msg delivery to sw appl or web service with choosen level of QOS.

Scanned by CamScanner

* WS - Security -
WS - Security specifies protection of SOAP msg through msg integrity, confidentiality & msg authentication. WS-Security also specifies how to associate a security with msg. XML-encryption & XML-signature used for WS-Security

* WS-Transaction - A WS-Transaction - a web services specification developed by IBM. The WS-Transaction specification describes co-ordination types that are used with the extensible co-ordination framework.

* WS-Coordination - The WS-cordination is a web services specification developed by IBM & Microsoft. It describes an extensible framework for providing protocols that cordinate the action of distributed applications. Such cordination protocols are used to support a no. of applications.

protocols for coordination

* WS-Context is a web-service specification developed by OASIS Corganization). Its purpose is to provide a means to reference a shared context, which relate a set of interation between web services. This context provides details of the application specific execution environment for these services. Context can be passed by value or by reference in which case they are retrived using Context Manager Service

* Web-Service Communication:
The web-service communication protocol are used - RPC, XMLRPC, SOAP etc.
The Service-oriented communication (SOC) is to enable communication through a Service-oriented architecture & to make communication as service.

Open Visual Studio .NET
On the file menu, click 'New' & then click 'Project'. Under 'Project types' click "Visual C# Projects", then click - ASP.NET Web Service under 'Templates'. Type 'MathService' in the location text box to change the default name (webservice1) to MathService.

3. Change the name of the default web service that is created from service1.asmx to MathService.asmx.
4. Click - Click here to switch to code view in the designer environment to switch to code view.
5. Define methods that encapsulates the funn of your service. Each method that will be exposed from the service must be flagged with a 'WebMethod' attribute in front of it. Without this attribute, the method will not be exposed from the service.
[NOTE - 'WebMethod' is useful to hide some implementation details called by public web service method.

Scanned by CamScanner

```
[web-Method]
public int Add (int a, int b)
{
    return (a+b).
}
```

Click - build on the build menue to build web services.
Browse the MathservicE.asmx web service page to test the web service.
If you set the local comp. to host the page - URL is -
http://localhost/Mathservice/Mathservice.asmx

JMS - Java Message Service.
Managing SOAP messages over the Java Message Service API can provide greater scalability & reliability than HTTP. JMS transport & stores web service request & response message in queue until they can be processed by the server.

SOAP is an XML-based messaging protocol. It defines set of rules for structuring message that can be used for simple one-way messaging but is particularly useful for performing RPC.
SOAP is a simple, lightweight XML protocol for exchanging structured & typed information on the web.
SOAP contains 4 parts -

① An extensible envelope expression - (mandatory)
- what features & services are represented in a message.
- Who should deal with them.
- whether they are optional or mandatory.

② A set of encoding rules for data (optional) -
- exchange instance of application-defined data types & directed graph.
- Uniform model for serializing abstract data model that can not be directly be expressed in XML schema.

③ A Convention for representation RPC (optional).
- How to make calls & response.

④ A protocol binding to HTTP & HTTP-EF (optional).

① SOAP Envelope - It describe the message identifying the contents & the envelope's processing information.

Scanned by CamScanner

② SOAP Transport - It defines the bindings for the underlying transport protocol such as HTTP & SMTP.

③ SOAP Encoding - It define a set of encoding rules for mapping the instance of the application - specific data types to XML - elements.

④ SOAP RPC - Conventions - It defines the representation of RPC req. & response. These SOAP Request & Responses are marshaled in a data type & passed in to a SOAP Body.

Basic Steps of Implementing Web Services.

① The service provider creates the web service typically as SOAP - based service interface for exposed business application. The provider then deploy them in a service container or using a SOAP runtime environment, & then makes them available for invokation over a N/w. The service provider also describes the web service as WSDL - based service description, which defines the clients & service container with a consis consistent way of identifying the service location, operation & its communication model.

② The service provider then registers the WSDL-based service description with a service broker, which is typically a UDDI registry.

③ The UDDI registry then stores the service description as binding templates & URLs to WSDL located in the service provider environment.

④ The service requestor then locates the required services by querying the UDDI registry. The service requestor obtains the binding information & the URLs to identify the service provider.

⑤ Using the binding information, the service requestor then invokes the service provider & then retrieves the WSDL service description for those registered services. Then the service requestor creates a client proxy application & establishes communication with the service provider using SOAP. SOAP

⑥ Finally the service Requestor communicates with the services provider & exchanges data or msgs by invoking the available service services in the service container.

Scanned by CamScanner

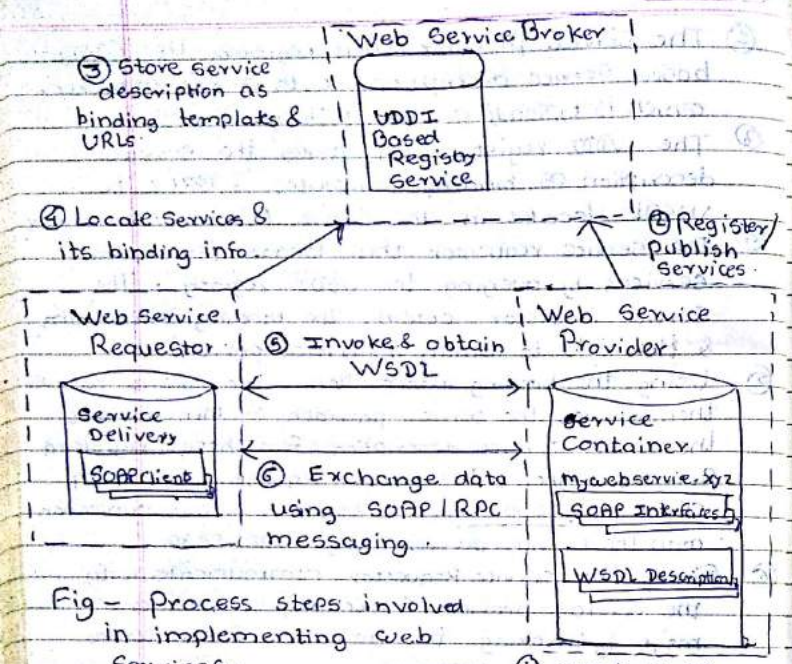


Fig - Process steps involved in implementing web Services.

* Developing Web Services -Enabled Application

The design & development process of creating a web services - enabled application is not different from the typical process of developing a distributed application.

In case of web services, it can be created as new application or from using an existing application by repurposing them as services.

In web services implementation, it is also possible to expose existing / legacy applin as services by encapsulating the core business functionalities of those underlaying application. The underlaying application can be of any application implemented in any programming language & running on any platform.

Following figure represents a typical web services implen model providing service - oriented interface supporting a varitey of back-end appln environment.

The implⁿ steps generally involved in developing web sservices solⁿ by exposing back-end business applⁿ are as follows -

① The potential business components of the underlaying applⁿ will be encapsulated as service - oriented interfaces using SOAP & then exposed as a web service by deploying them in a web services service container or a SOAP Runtime environment. Using the SOAP - based interfaces, the service container handle all the incoming SOAP request / responses or messageing - based operation & maps them as method & arguments of the underlaying business applⁿ.

② WSDL - based service description will be generated & then reside in a service container. WSDL defines the communication contract required for invoking the SOAP - based service interfaces. These WSDL - based service description will be published in a UDDI registry as service templates & its location URLs. The interfaces

Scanned by CamScanner

required for publishing in the UDDI registry are usually provided by the web service container provider.

③ The service requester finds the services using the discovery mechanism (registry API) & obtains the service description & its provider location URL. It then connects to the service provider to obtain WSDL.

④ To invoke the services exposed by the service provider, the service requester is required to implement SOAP-based client interfaces according to the service description defined in the WSDL.

These steps are usually common at all levels of web services development, such as irrespective of the target appln environment such as - J2EE, CORBA, Microsoft .NET.

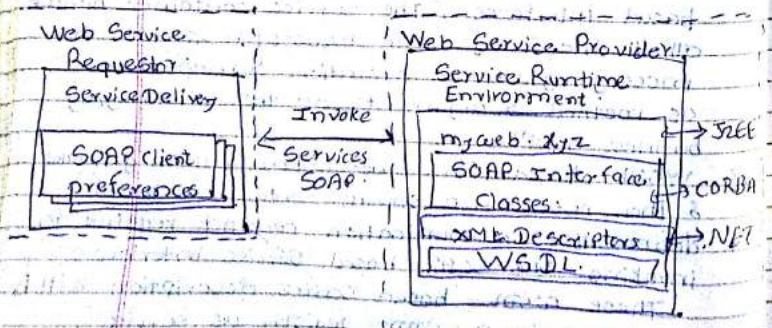


Fig - Exposing appln through Web Service

Chapter-3

SOAP. Simple Object Access Protocol

Inter-application Communication - IAC is used to address the critical resource allocation & access for both synchronous & asynchronous processing & to guarantee resources & appln

SOAP as a Messaging Protocol

SOAP is the standard messaging protocol used by web services. SOAP's primary application is inter application communication. SOAP covers the following main areas -

- A message format for one-way communication describing how a message can be packed into an XML document.
- A description of how SOAP message should be transported using HTTP (for web based interaction) or SMTP (for email-based interaction).
- A set of rules that must be followed when processing a SOAP message & a simple classification of the entities involved in processing a SOAP message.
- A set of conventions on how to turn an RPC call into a SOAP msg & back.

SOAP is a way for a Program running in one operating system to communicate with a program running in either the same or a different operating system, using HTTP or any other transport protocol & XML.

* Structure of SOAP Message :-

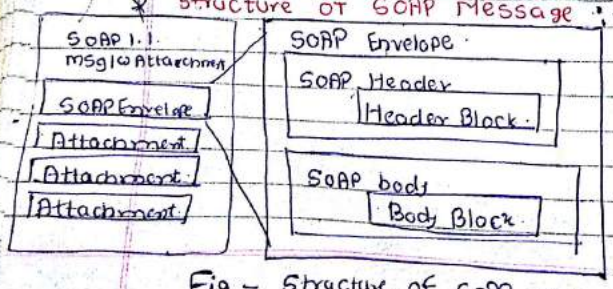


Fig - Structure of SOAP msg.

The structure format of a SOAP message contains the following elements.

1. Envelope
2. Header (optional).
3. Body
4. Attachments (optional).

The above figure represents the structure of a SOAP message with attachments. SOAP msg is represented by a SOAP envelope with zero or more attachments. The SOAP message envelope contains the header & body of the message.

* The SOAP message attachments enables the message to contain data, which include XML & non XML data (like text / binary files).

* The SOAP message package is constructed using the MIME.

1. SOAP Envelope - The SOAP envelope is the primary container of a SOAP messages structure & is the mandatory element of a SOAP message.
It is represented as the root element of the message as 'envelope'.
Every envelope element must contain exactly one body element.
If an Envelope contains a Header element, it must contain no more than than one, & it must appear as the first child of the envelope, before the the body.

Scanned by CamScanner

The SOAP Envelope SOAP-ENV: Envelope as its primary root element & it relies on defined "XML Namespace" commonly identified with a keyword Xmlns

xmlns

The SOAP message requires defining two basic namespaces -
① SOAP Envelope ② SOAP Encoding

Header :-

* The SOAP header is presented as the first immediate child element of a SOAP envelope.

It also contain zero or more optional child elements, which referred to as SOAP header entries block.

The SOAP encodingStyle attribute will be used to define the encoding of the data types used in header element entries.

The SOAPactor attribute & SOAP mustUnderstand attribute can be used to indicate the target SOAP application node. (Sender / Receiver / Intermediary) & to process the Header entries.

```
<SOAP-Env:Header>
<wiley:transaction
xmlns:wiles="http://jws.
SOAP-ENV:mustUnderstand="1">
<KeyValue> 5 </kv>
</wile:Tr>
</SOAP-ENV:Header>
```

* Body :-

SOAP envelope contains a SOAP body as it child element, it may contain one or more optional SOAP body block entries. The Body represents the mandatory processing information. A Body block of a SOAP message can contain any of the following.

* RPC method & its parameters.
* Target application (receiver) specific data.
* SOAP fault for reporting errors & status information.

In general, the SOAP Body can contain information defining an RPC call, business document in XML & any XML data required to be part of the message during communication.

Body is a mandatory element which envelopes the message to send in XML format. This element holds either the requested data (Response) or an error message (fault). Requested data represent the application-specific data exchanged with a web service. This information can be XML data or parameters to a method call. Inside the SOAP <body>, the method call information as well as its related argument are defined, the response to a method call is placed & error information can be saved.

* Fault - is optional element that provide inform about error that occured while processing the msg.

Scanned by CamScanner

The SOAP Fault element is used to handle error & to find out status information.

It provide following elements:

1. * FaultCode - The Fault Code element defines the algorithmic mechanism for the SOAP application to identify the fault. It contains standard value for identifying the error or status of the SOAP application.

The following fault code element values are defined in the SOAP 1.1.

- VersionMismatch - This value indicate that an invalid nam namespace is defined in the SOAP envelope or an unsupported version of a SOAP msg.

- MustUnderstand - This value is returned if the SOAP receiver node cannot handle & recognize the SOAP header block when the MU attribute is set to 1 (0 for false & 1 for true).

- Client - ~~in~~ This faultcode indicates when a problem originates from the receiving client. The possible problem could vary from an incorrect SOAP message, a missing element or incorrect name space definination.

- Server - This faultCode indicates that a problem has been encountered during processing on the server side of appln.

2. * FaultString - The FS element provide a readable description of the SOAP Fault.

3. * FaultActor - The FaultActor element provide the information about the SOAP Actor (Sender / Receiver / Intermediary). In the message who is responsible for the SOAP fault at the particular destination of a msg.

4. * Detail - D. element provides the appln-specific error or status information related to the defined body block.

4) SOAP - Attachment.

The SOAP msg can contain the SOAP-Attachments in any data ~~fo fo the format~~ that can be ASCII or binary (such as XML or non-text).

SOAP att. are not part of SOAP envelope but related to SOAP message.

To add file to a SOAP message a tag is added within the body of the message.

<attachment href = "{URI}" />

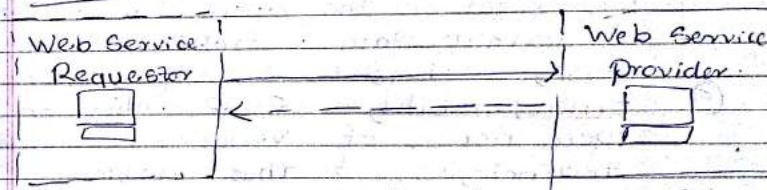
Scanned by CamScanner

* SOAP communication model:-
The SOAP communication model describe how to invoke web service & relies on SOAP. The SOAP communication model is defined by it communication style & its encoding style. SOAP supports two possible communication style (1) RPC & (2) document (msg).

*(1) RPC-style Web Service:-
RPC-style web service are used as remote objects on the client appl^n side.
Client send their request as a method call. The method returns a response message. This information is formatted as sets of xml element loaded into SOAP msg.
Asynchronous & loosely coupled

2) * Document (msg)-style web Service
SOAP supports documents to the provider instead of sending a set of arguments.
The messaging-based comm^n model defines a loosely coupled & document driven communication.
The service Requestor invoking a messaging-based service provider does not wait for a response. The figure represents a mess-based comm^n model in web service architecture

The client service requestor invokes a mess-based web service; it typically send an entire document rather than sending a set of parameters. The service provider recieves the document, Process it, & then may or may not return a message. Depending upon the implementation, the client can either send or receive a document asynchronously to & from a messaging-based service, but it can not do both functionalities at an instant.



Asynchronous means that the client which invokes a web service, does not or can not wait for a response. Thus asynchronous is a one-way operation. The client send a request in the form of an xml msg. Provider receives the msg & Process it & send the result when processing is completing

Advantages of SOAP.

① Simplicity : SOAP is simple as its uses XML that is well structured & easy to parse.

② Portability - SOAP is platform independent & thus portable.

③ Firewall - Friendliness - SOAP is capable of getting past firewall which are totally blocking for other protocol. This is possible due to using the HTTP protocol.

④ Use of open standard - SOAP is base based on the open standard XML to format data. SOAP becomes easily extensible & well supported.

⑤ Interoperability - SOAP relies on open instead of vendor-specification technologies & thus enables distributed interoperability & loosely coupled application.

Disadvantages

① SOAP is stateless which implies that the requesting appl^n has to reintroduce itself to other appl^n if add^n conn^n are needed as if it was connected for the first time.

② SOAP serializes by value & not by reference.

③ HTTP is used to transport the messages. This imposed a request/response architecture which did not suite all situation. As HTTP is relatively slow. So the performance of SOAP is affected.

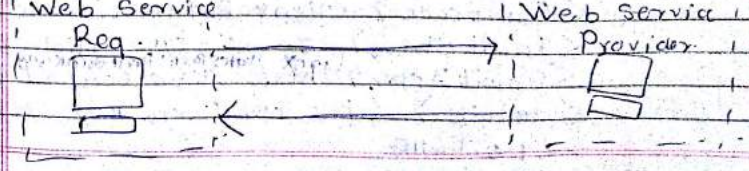
Tightly Coupled & Synchronous

① * RPC- Based Communication Model:

The RPC-based communication model defines a request/response-based synchronous communication. When the client send a request, the client wait until a response is sent back from the server before continuing any operation. The RPC-based web service are tightly coupled & are implemented with remote objects to the client appl^n.

The fig. represents the RPC-based comm^n model in web service.

The client have a capability to provide parameters in method calls to the web service provider. Then clients invokes the web service by sending parameter values to the web service provider that executes the required methods & then send back the return value. Additionally, using a RPC-based comm^n model, both the service provider & requestor can register & discover services respectively.



Scanned by CamScanner

Synchronous means that every time a client access a web service application, the client receives the SOAP response.
Synchronous is request-response operation.

Error Handling in SOAP

The API throws a SOAP Exception error. The error code can be obtained from the <detail> property tag of the SOAP Exception.

SOAP Exception Example.

Fault code Namespace :
Fault code Name : Server
SOAP Actor that threw Exception :
Error msg -
Detail :

```
<S: Envelope xmlns:S = " ">
<S: Body>
  <S: Fault>
<faultcode> Client Authentication </fc>
<faultstring> Invalid credentials </fs>
<FaultActor> http://acme.com/Partner/Security/Unix-states/Town : Partner Security </fa>
<details> <!-- appln details --> </d>
</S: Fault>
</S: Body>
</S: Envelope>
```

ERROR Code

1. Invalid Login
2. Invalid token
ZipCode Database error
Illegal zipcode
No zipcode
Session expired
Illegal city / state

Exception thrown by a web service method created using ASP.NET are sent back to the client in the form of a SOAP fault.
A SOAP fault is a fault xml element within a SOAP msg that specifies when an error occured. It may contain details such as the exception string & source of the exception.
Exception is any error condition or unexpected behavior encounterd by an executing program. Exc. can occur due to a no. of reasons such as fault in your code, operating system, resource not available, unexpected condition & so on.
While your applⁿ can recover from some of these conditions, most of the runtime exceptions are irrecoverable. In that case, you need an effective way of handling those exceptions & informing the callers of the exceptions of the same.
Handling exception in a web service is no different than handling exception in web or windows applⁿ.

Scanned by CamScanner

You need to be aware of the fact that u need to communicate the exception information to the consumer of your web service in a platform independent manner based on the SOAP specification. To accomplish this, you should use the SoapExcep^n class that abstracts the complexities of the SOAP fault creation process.

The SoapException class consist of the SOAP following properties that need to be populated before throwing the exception to the consumers.

* Message - content of the exception.
* Code - Specifies the Fault Code (ex Client Fault Code, Server Fault Code)
* Actor - URL of the web service where the exception has occured.
* Detail - used to communicate max information about the exception to the caller.

Namespace - System.Web.Service.protocol.
SoapException.

The SOAP Exception type exposes the following members

Constructors -

SOAPException () - Initilizes a new instance of the SoapException class.

SoapException (SerializationInfo, StreamingContext) - initilizes new instance of the SOAPException class with serialized data.

Namespace - A logical grouping of the names used within a prog.

properties -

Actor -

Code -

HelpLink - Get or set a link to the help file associated with this exception.

Methods -

Equals (Object) - Determine whether the specified obj is equal to the current Object.

get cause(), getMessage, initcause()

return throwable obj — get detailed message of this exception — initialise the cause field of this exception to given throwable obj

```
try
{

}
catch (      )
{
}
```

```
SoapException se = new SoapException
("Fault Occured", SoapFaultCode.
ClientFaultCode);
throw se;
```

Within a Try/Catch block, catch the SoapException (Any exception thrown by an xml web service method are thrown as a SoapException.

Scanned by CamScanner

```
[WebMethod]
public void GetException()
{ try {
        int i = 1;
        int j = 0;
        int z;
        z = i / j;   // Generate exception
  }
  catch ( Exception ex)
  {
  throw RaiseException (ex.Message,
    ex.Source );
  }
}
// Create the SoapException obj with all
the error details.

public SoapException RaiseException
  (String uri, String errormsg,
   String Source )

  Catch (SoapException se)
  { String msg = se.Message;
    String show = se.Detail.outerHtml;
```

Annotation - metadata / Specific part of original data / Marking

Building web services with JAX-WS

JAX-WS stand for Java API for XML Web Service. JAX-WS is a technology for building web services & clients that communicate using XML. JAX-WS allows developers to write msg-oriented as well as RPC-oriented web services.

In JAX-WS, a web service operation invocation is represented by an XML-based protocol such as SOAP.

JAX-WS is platform independence of the Java Programming language.

JAX-WS uses technologies defined by the (W3C) World Wide Web Consortium. It uses HTTP, SOAP, WSDL.

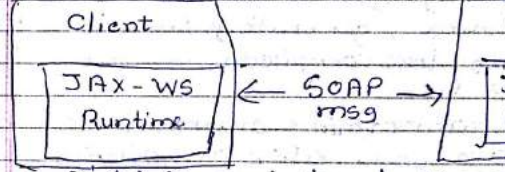


fig: Communication betn a JAX-WS webservice & a client

The starting point for developing a JAX-WS web service is a java class annotated with the javax.jws.webservice annotation.

The @WebService annotation defines the class as a web service endpoint.

Scanned by CamScanner

Annotation are used to provide meta data for your java code.

Java has 3 built in annotation that you can use to give instruction to the java compiler.

Java annotations are used for following purpose.

1. Compiler instruction
2. Build - time instruction.
3. Runtime instruction.

Java annotation can be used at build - time, when you build your s/w project. The build process includes generating source code, compiling the source, generating xml files, packging the compiled code & file into a JAR files.

Java annotations are not present in your Java code after compilation.

@ - character signals to the compiler that this is an annotation.

@webservice - is name of annotation

A service endpoint interface (SEI) is a Java interface that declares the methods that a client can invoke on the service. An SEI is not required when building a JAX - WS endpoint.

The web service implementation class implicitly defines a SEI.

There are the basic steps for creating the web service & client

1. Code the implementation class.
2. Compile the implementation class.
3. Deploy the WAR file.
4. Code the client class.
5. Use wsimport to generate & compile the stub file.
6. Compile the client class.
7. Run the client.

```
import java.jws.webService;
@webService ( portName = ,
  serviceName = TargetNamespace = ,
  endpointInterface )

public class Calculator implement
  CalculatorWS {
public int sum (int a, int b)
  {
    return a+b;
  }
public int multiply (int p, int q)
  {
    return p*q;
  }
```

Scanned by CamScanner

@ web Service Endpoint interface

```
import javax.jws.webservice;
```

service endpoint interface

```
@ webservice (targetNamespace = "")
public interface CalculatorWS {

    Public int sum(int a, int b);
    public int multiply (int p, int q);
}
```

That can make testing & using the web service from other Java client far easier.

```
import javax.jws.WS;
import javax.jws.WebMethod;

@ webservice
public class Hello {

Private String msg = new String("Hello");
public void Hello()
    {
    }
@ webmethod
public String SayHello(String name)
    {
    return msg + name + ".";
}
}
```

Step 1

Open Netbeans - Select file - New Proj → Java Web - Web Applh.

Click Next - Enter project name - " Hello WebServic "
Next - Select Server (Glassfish Ser.

Click finish.

Step 2 - Create WS.

Right Click on the proj - >> new - web service - Enter web service name - Enter package -

⊙ - Create web service from scratch.
& click on - finish.

Adding an operation to the web service
The netBean IDE provide a dialog for adding an operation the operation to a web service

click on the - Designer View.
Add Operation. - give name to the operation
Add parameter names & par. type
& click ok.
OR
Find the web service node in the Project Window. Right Click that node & - add operation.

Scanned by CamScanner

Save Specifications

ex -

parameter name	type
i	int
j	int

Click Source & view the code

```
package cal ;   → package name
import javax.jws.WebMethod;
import javax.jws.WebParam;
        .WebService
@WebService ( )
public class NewWebService - webservice name
{
@WebMethod (operationName="Add")
public int Add (@WebParam (name="i")
int i ,@WebParam (Param name="j")
int j )
{
    int k = i+j;
    return k;
}
}
```

(name space)

* Deploying & Testing the Web Service

Right Click the Project & Choose Deploy. The IDE starts the appln server, builds the appln & deploy the appln to the server.

In the IDE's Project tab, expand the Web service node of your projectname. Right click on - Projectname & choose Test Web Service.

The IDE open the Tester page in your browser, if u deployed a web appln to the Glass Fish Server. For the Tomcat server & deployment of EJB module, the situation is different.

@webParam (name="j")

'name' specifies the name of parameter.

partName = Specifies the value of name attribute.

Scanned by CamScanner

Assignment No - 3

1) SOAP as a messaging protocol comment
2) Explain the structure of SOAP message.
3) What is mean by SOAP communication model & Explain its types.
4) Explain error handing in SOAP.
5) Explain building web services using java.
6) What are advantages & disadvantages of SOAP?

Ch-04

Describing & Discovering Web Services

* WSDL in the world of Web service.

The web service description language is an Xml - based language that is used for describing the functionality offered by a web service. A WSDL description of a web service (also reffered as a WSDL file) provides a machine readable description of how the service can be called, what parameters it expects, what data structure it returns.

* WSDL is a specification defining how to describe web service in a common XML grammer. WSDL describes 4 critical pieces of data.

* Interface information - describing all publicly available functions.
* Data type information for all msg req & msg respor responses
* Binding information about the transport protocol to be used.
* Address information for locating the specified service.

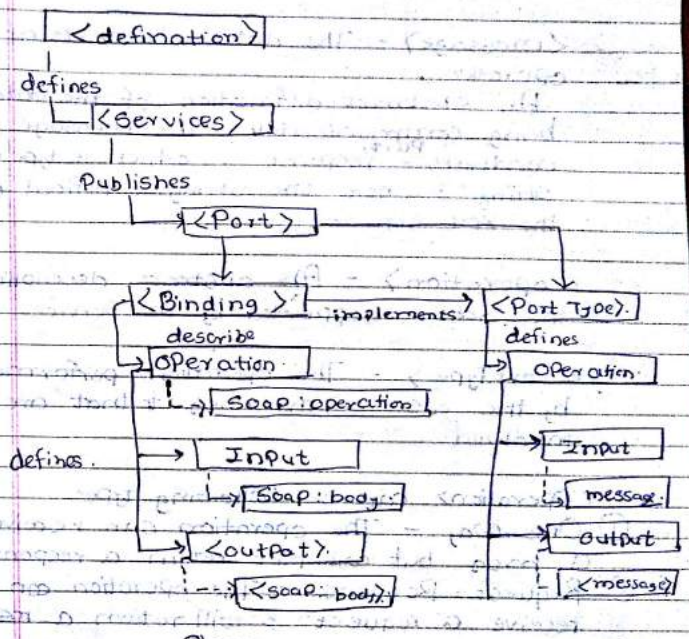
In a nutshell, WSDL represents a contract between the service req & service provider. in much same way that a Java interface represents a contract between client code & the actual Java object.
The difference is that WSDL is language & platform independent.

Scanned by CamScanner

SOAP based service interface is running at service provider side.
& used primarily to describe SOAP Services.

Using WSDL, a client can locate a web service & invoke any of its publicly available function.

* Structure of WSDL Interface



* Abstract Part - describes -
* the message it sends & receives
* The msg operation associates a msg exchange pattern with one or more message.

① <types> - The data types used by the web service (XML schema is used in general).

Scanned by CamScanner

Provides information about any complex data types used in the WSDL document:

(2) <message> - The data elements of an operation.
An abstract defination of the data being communicated. The message contains - part, response, which is type of string & where the string is defined by the XML Schema.

(3) <operation> - An abstract description of the action supported by the service.

(4) <PortType> - The operation performed by the WS & the msg that are involved.

<Operation> can be following types.
① One Way - The operation can receive a msg but will not return a response
② Request - Response - The operation can receive a request & will return a response.
③ Solicit - response - The operation can send a request & will wait for a response
④ Notification - The operation can send a msg but will not wait for a response.

For performing operation which concrete protocol & data type req.

* Concrete Part

(1) <Bindings> - describe how the operation invoked by specifying concrete protocol & data format specification for the operation & msg.
① - binding attributes are -
* name (Any name)
* type (points to the port type defined in the abstract part)
② Soap: binding attributes are -
* Style ("rpc" / "document")
* Transport (communication protocol - SOAP & Transport Protocol - HTTP/SMTP
③ Operation Element -
* defines each operation that the port expose.
* must also specify how the Input & output are encoded
[literal (for document) & SOAP (for rpc)]
- Specifies bindings of each operation in the PortTypes section.

(2) Service & Port -
* A service may have more than one endpoint, with each one defined by its own port element.
* The port element corresponds to a particular binding & includes information on how to access it (URI)
* Different ports represent different binding for the same port type.

Scanned by CamScanner

- allows the same fun functionality to be accessible via multiple transport protocols & interation style.

Service specifies the port address of the binding. The service is a collection of n/w endpoints or ports.
service consist multiple <port> elements
Port - specifies a single endpoint as an address for the binding, thus defining a single communication endpoint.

ex - Message

```
<message name = "ordermsg" >
  <port name = "Product name"
    type = "xs : string" />
  <port name = "quantity" type = "xs : integer" />
</message>
```

responce

ex - operation & port

```
<PortTypename = "port type" >
  <operation name = "order" >
    <input message = "ordermsg" />
  </operation>
</PortType>
```

ex - binding

```
<binding name = "Procurement soap binding"
  type = "tns : procurement PortType" >
  <soap : binding style = "document"
    transport = "http://schemas.xmlsoap.org/soap/http" />
```

name of interface

```
  <operation name = "order" >
    <soap : operation soapAction = "http://example.com/order" />
    <input>
      <soap : body use = "literal" />
    </input>
    <output>
      <soap : body use = "literal" />
    </output>
  </operation>
</binding>
```

```
<service name = "Procurement Service" >
  <port name = "Procurement port"
    binding = "tns : Procurement Soap binding" >
    <soap : address location = "http://example.com/procurement" />
  </port>
</service>
</defination>
```

reference to binding

N/w address of WS

* WSDL document structure

```
<defination>
  <types>
    defination of types ...
  </types>
  <message>
    defination of message
  </message>
  <PortType>
    defination of a port
  </PortType>
```

Scanned by CamScanner

<binding>
defination of bir binding
</binding>
<Service>
defination of service
</Service>
</defination>

Assignment No-2

WSDL Life Cycle

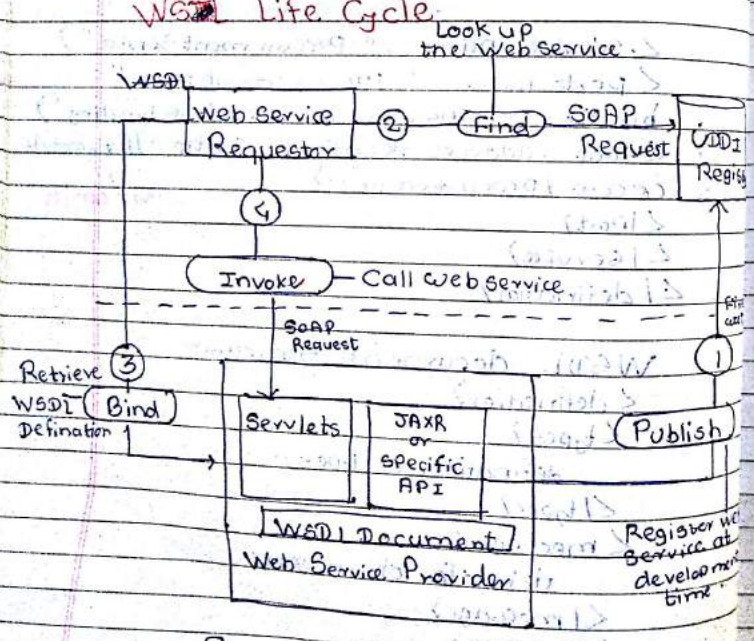


Fig - Web Services Lifecycle.

① Step-1 :-
Step-1 illustrates a service provider publishing its web service to a UDDI registry. This is when the service provider would create a WSDL defination & publish a link to this defination along with the rest of the web service information to a UDDI registry.

② Step-2 illustrates an interested service user locating the web service & finally obtaining information about invoking the web service from the published WSDL defination. This step involves downloading a WSDL defination to the service user system & deserializing WSDL to a java class (or any other language). This java interface serves as a proxy to the actual web service. It consist of the binding information of the web service.

③ Step-3 shows the service user binding at runtime to the web service. In this step, the service user's application would make use of the java interface representing WSDL as a proxy, in order to bind to the web service.

④ Step-4 finally shows the service user invoking the web service based on the service invokation information it extracted from the web service WSDL defination. This is when the service user's application would make use of the java interface representing WSDL as a proxy in order to invoke the methods/functions exposed by the web service.

Scanned by CamScanner

< defination> Element

The <definations> element specifies the name of the document in which this WSDL defination is stored.

This element specifies namespaces that would be used in the rest of the WSDL document.

The following are the two important namespaces that the <definations> element defines -

① WSDL instance specific namespaces —

The TargetNamespace attribute of the <defination> element lets the WSDL document make th make the reference to itself as an XML schema namespace.

② Default namespace for a WSDL document:

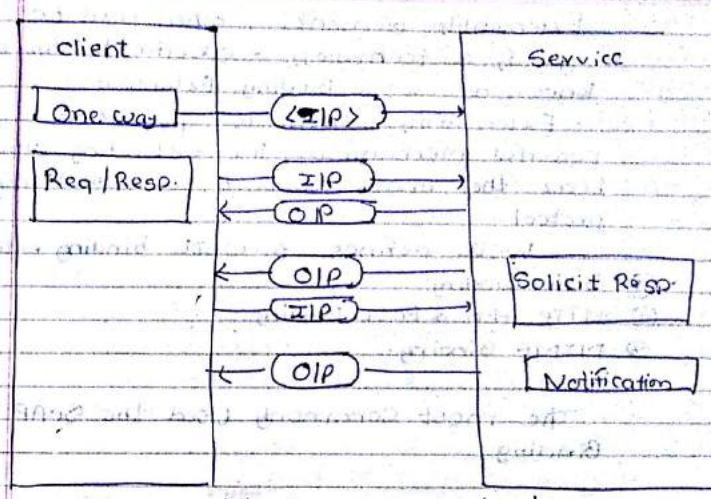
The default namespace is specified by xmlns = "http://schemas.xmlsoap.org/wsdl/". The default namespace indicates that all of the element in this WSDL defination without a namespace prefix, such as <types>, <message> & <portType> are part of this namespace.

one-way - The endpoint receives a msg.

R-R - The endpoint receives a msg & send correlated msg.

Solicit-Response - The endpoint send a msg & receives a correlated msg.

Notification - The endpoint sends a msg.



WSDL binding refers to how your

* WSDL Bindings - WS is bound to protocol

WSDL binding referes ~~referes~~ to the process of associating protocols or data format information with an abstract entity such as <message> <operation> or <portype>.

Binding defines a message format & protocol defines ~~details~~ the details for operations & msgs defined by a particular portType.

The name attribute provides a unique name among all binding binding defined within in the enclosing WSDL document.

* WSDL Binding Extensions -

WSDL allows user-defined elements also known as Extensibility elements under various elements defined by a default WSDL namespace.

Scanned by CamScanner

Extensibility elements, when used to specify a technology - specific binding are known as WSDL Binding Extension.

Extensibility elements provide a powerful mechanism for extending WSDL. becoz they enable support for N/W & msg protocol.

WSDL defines 3 WSDL binding extension

1. SOAP Binding
2. HTTP GET & POST Binding
3. MIME Binding.

The most commonly used the SOAP Binding.

WSDL binding support ~~for~~ four operation -

All 4 types of operation supported by WSDL - one way, req-resp, solicit resp., notification - represents an abstract notion only. Binding describes the concrete correlation to these abstract notions. Binding determines how the msgs are actually sent within the single communication. Thus binding for a specific operation type must be defined in order to successfully carry out that type of operation.

Note- The Although the WSDL structure supports the bindings for these 4 operations, the WSDL specification defines bindings for only one way & Req-Resp. operation. So in order to use solicit Resp & notification operation, the commⁿ protocol must define WSDL binding extension.

SOAP Binding

A web service SOAP binding is a protocol that describes how message between web service provider & requestors are defined & transferred.

This binding provides the following SOAP protocol specific information.

* An indication that the binding is bound to the SOAP 1.1 protocol.
* A way of specifying an address for a SOAP 1.1 protocol.
* The URI for the SOAP action HTTP header for the HTTP binding of SOAP.
* A list of defination of header that are transmitted as part of the SOAP envelope.

(i) <Soap: binding> - The <soap: binding> element says that the binding is bound to the SOAP protocol format, i.e. envelope, header & body.

(ii) Style - The style attribute indicates whether the operation supported by this bindings are RPC-oriented or Document-oriented.

In RPC-oriented communication, the msgs contains parameter & return values.

In Document-oriented commⁿ, the msg contains documents.

The style of commⁿ can be useful becoz it helps in selecting the programming model for communicating with the web service.

ex- If a web service is described to support RPC, we can choose a JAX-RPC programming model to communicate with it.

If a web service is described to support

Scanned by CamScanner

the document style comm^n, we can choose a JAXM programming model.

(b) transport - The transport attribute specifies the transport binding for the SOAP protocol. The URI value of http://schemas.xmlsoap.org/soap/http corresponds to the HTTP binding as defined in the SOAP specification.

The respective URIs can be used to indicate the other type of transports such as SMTP / FTP. Transport specifies the comm^n protocol that is used to transport msg.

ex - <Soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

(2) <Soap:operation> - Provide info^n for operation.

The <Soap:operation> element defines the information with regard to communication style & the soap action header at that specific operation level.

The "SoapAction" attribute specifies the value of a soap action header in the form of a URI.

ex - <Soap:operation name="order">
<Soap:operation SoapAction="http://example.com/order"/>

(a) SoapAction attribute specifies the value of the SoapAction in the HTTP header for this operation.

(3) <Soap:body> - This element defines how the msg <Part> element appear inside the SOAP body element. Based on the style of communication, RPC / Document the <body> element of the SOAP msg is constructed.

(a) Use - The use attribute indicates whether the msg <Part> elements are encoded using some encoding rules or whether the <Part> elements already defines the concrete schema of the msg.

<Soap:body use="literal"/>

This element enables you to specify the details of the input & output msg.

(b) encoding style - Indicates a particular encoding style to use. It is optional.

(c) namespaces - Indicates the namespaces.

The WSDL binding element describes how your web service is bound to a protocol. In other words, how your web service is accessible. To be accessible, the web service must be reachable using some N/W protocol. This is called "binding" the web service to the protocol.

Scanned by CamScanner

* The binding name is referenced by the service element. Thus it should be unique within the WSDL file.
* The interface attribute should refer to the name of interface element defined in the WSDL file.
  Hence the tns: prefix
  (in this WSDL's target Name Space)
* The type attribute tells what kind of message format the interface is bound to.
  In this given example type (msg format is SOAP).

* The wsoap: protocol - attribute specifies a SOAP binding - in other words, how the msg is transported.
  The value in the in the example specifies HTTP.

* The "operation" element references an operation defined in the interfac. This reference is made by - "ref" attribute.
* The wsoap: mep - attribute of the operation element, specifies the "Message Exchange Pattern".

If use = "literal" - then the msg parts should be used as are referring to concrete schema defination & no encoding is needed.

If use = "encoded" then the msg parts are referring to abstract schema defination & concrete msg can be produced produced by applying the specified encoding style.

encoding Style = "uri-list" - indicates encoding style to be used in case of use = "encoded".
namespace = "Uri" indicates the namespace to be used in case of use = "encoded".

* WSDL Tools
  WSDL tools typically provide functionality in terms of the following.
  1. WSDL Generation - generating WSDL from an existing service component. Ex - J2EE component or a Java Bean Component or from scratch.
  2. WSDL Compilation - A typical WSDL compiler would generate the necessary data structure for the implementation of the service. The generated impl^n skeleton contains all the methods (operation) that are described in the given WSDL defination.
  3. WSDL Proxy Generation - This functionality can read a WSDL & produce a specific language binding consisting of all the code required to bind the web service & to invoke the web service.

Scanned by CamScanner

service functions at runtime.
This functionality is typically used at the client side.

WSDL Tool
1. Sun One Studio 4.
2. Systinet WASP.
3. The Mind Electric Glue.
4. IBM Web Service Toolkit.
5. BEA Weblogic Workshop.
6. Apache Axis.

Port Element.

The only purpose of the 'port' element is to specify the N/w address of the endpoint hosting the WS.
'Port' is a sisingle end-point defined as a combination of a binding & a N/w address.

The Soap:address element is used to give a port an address

Service Element.

A service is a collection of ports. Although a WSDL document can contain a collection of 'service' element, by conventions a WSDL document contain a single service.

```
< service name = " Temp" >
< Port
binding = " Temp. Soap.binding"
name = " Temp port" >

< soap: address
  location: // example. com //>
</ Port>

< Port > - - - < /Port>
</services>
```

Scanned by CamScanner

Limitations of WSDL:

Its incapability of being able to describe complex business WS, which typically are constituted by orchestrating multiple finer grained WS.

This drawback is due to the lack of support for workflow description in WSDL.

To overcome these limitation of WSDL, standards such as ebXML Collaborative protocol profile / Collaborative Protocol Agreement [CCP/A], WS choreography interface (WS(I)) can be leveraged.

Though a WSDL provides details on how to invoke a service, it does not contain information about the use of the service. Programmer should be aware about the use of a service.

Service Discovery

Service discovery is a process of locating web service providers & retrieving WS description that have been previously published.

By querying the service registry for web services matching the needs of a service requestor.

- A query consist of search criteria such as-

* A type of the desired service, preferred price & maximum no. of returned results & is executed against service information published by service provider.
* Discovering web service is a process that is also dependent on the architecture of the service registry.

After discovery process is complete, the service developer or client appl^n should know the exact location of web service (URI), its capabilities, & how to interface with it.

Benifits of service discovery.

1. Ability to optimize performance, functionality & cost.
2. Easier introduction of system upgrade.

Scanned by CamScanner

UDDI

- UDDI is an XML based standard for describing, publishing & finding web services.
- UDDI stands for - Universal Description, Discovery & Integration.
- UDDI is a specification for a distributed registry of WS. UDDI is platform independent & open framework.
- UDDI can communicate via SOAP, CORBA, Java RMI protocol.
- UDDI uses WSDL to describe interfaces to the web services
- UDDI has two parts -
  * A registry of all a web service's metadata including a pointer to the WSDL description of a service
  * A set of WSDL port type definations for manipulating & searching that registry
- UDDI is (UDDI 1.0) was originally announced by Microsoft, IBM & Ariba in Sept. 2000
- UDDI is a directory service where business can regist register & search for WS
- Store information about WS.

UDDI Regis

* An implementation of the UDDI specification is termed as a UDDI registry.
  UDDI registry services are a set of services that provide access t access to the UDDI registry. Registry services can perform a activities - authenticating & authorizing reg. registry requests etc.

* UDDI Benefits -

Before UDDI, there was no internet standards for buiseness to reach their customers & partners with information about their products & services. There was no any method of how to integrate system.

UDDI can help to solve -

1. Making it possible to discover t the right buisiness from the millions currently online
2. Reaching new customer
3. Describing services & business process

UDDI Registryes -

Public & Private UDDI Registries

A UDDI registry can be operated in two modes - public Mode & Private Mode.

Public UDDI Registries - A public UDDI registry is available for everyone to publish / query the business & service information at on the internet.

A public registry is a collections of peer directories that contain infor^n about business & services.

For a list of public registries, refer to http://www.uddi.org/register.html.

IBM maintains two public reg called IBM UDDI Business Registry & IBM UDDI Test Registry.

Scanned by CamScanner

The IBM UDDI Test Registry is used to test & validate your web service.

Business Registry - is a public impl^n of the UDDI standards that represented a master directory of publicaly available

Private UDDI - A private registry allows you to publish & test your internal application in a secure, private environment.

IBM has developed a private registry called the IBM WebSphere UDDI registry. It is a persistent storage mechanism for information published in the Registry. Many companies choose private UDDI registries for - control of access to the information, control over updating the information, reliability of the content of the registry.

Use of UDDI Registry

A business or company can register 3 types of information into a UDDI registry. These information is contained into 3 elements of UDDI.

These 3 elements are -

1. White Pages
2. Yellow Pages
3. Green Pages

1. White Pages

This category contains.

Basic information about the comp & its buisness.

Basic contact info,n including b. name, address, contact phone no etc.

Unique identifier for the company tax IDs. This infor,n allows others to discover your ws based upon your business identification.

It identifies organizations (companies or more likely departments).

2. Yellow Pages -

This category contains -

- This has more details about the company & includes description of the kind of electronic capabilities the company can offer to anyone who wants to do buisness with it.
- It uses commonly accepted industrial categorization shemas, industry code, product codes, business identification codes & the like to me make it easier for companies to search through the listing & find exactly what they want.
- It contains detailed business data.
- It categorize services.

Scanned by CamScanner

③ Green Pages -

This category contains

- Technical information about web service
- This allows someone to bind to a WS after it's been found.
- The various interfaces.
- The URL Location.
- Discovery information.
- information about a compan's key business process, such as operating system, supported programs, purchasing method, billing requirements & other higher level business protocol.

UDDI Data Structures.

UDDI support the following 4 core data structure:

① Buisness Entity
② Buisness Service
③ Binding Template
④ TModel

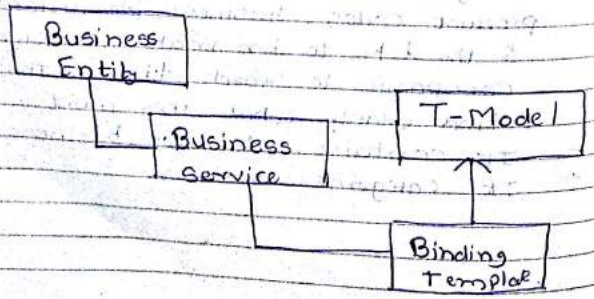


Fig - UDDI containment.

① Business Entity -

BE is the topmost entity container in a UDDI data structure. BE contains information about the business that has published the service such as - Business name, description, contacts & identifier. BE represents the provider of WS.

① Name & descriptions - The business entity can have a set of names & descriptions, in variety of languages (if necessary).

② Contacts - The list of people who are associated with the business entity. A contanct can include - for example, a contact name, address, phone no.

③ Categories - Set of categories that represent the business entity's feature or quantities.

④ Identifiers - The BE can be associated with unique identifier no. for uniquely identify it.

⑤ Discovery URLs are additional links to document describing the BE.

② Business Service

The BS structure represents an individual WS provided by business entity. Its description includes information on how to bind to the WS, what type of WS. BS represents a collection of WS offered by BE. Every business service has a unique service key for identification.

① Name & Description - The business service can have a set of names & description, in a variety of languages.

Scanned by CamScanner

Categories - A set of ca categorie categories that represent the business service feature & quatities.

ex - The business service can be associated by a category that represents service availability version etc.

Business Service is ~~set of~~ logical group of one or more web services in a single business Entity.

Binding Template -

Binding Template is used to bind web services & applications by providing the necessary technical ~~inf~~ ~~inform~~ information about Web Service. A binding Template represents a single web service that has an access point to the Web Service.

Every binding template is a child of a Business Service.

Every binding template has a unique bindingKey for identification.

An access point provides a routing information such as - URL, email address, telephone no to locate the service.

An access point represents the service endpoint.

The bindingTemplate represents the actal implementation of the Web Service.

Categories - BT can be associated with categories to reference specific feature of the binding template

ex - Certification Status or versions.

T-Model -

The T-Model provides technical information for a Web Service. It contains URL addres of the WS, metadata for the technical Infor information, unique key for identification.

T-Model stands for Technical Model.

A T-Model is a way of describing the various business, service, template struct stored within a UDDI registry.

Note - The technical information about WS is not stored in a UDDI registry. TModel provides URL address to locate the information.

TModel are described by the following element

1. Name & Description - —"—
2. An overview document is a reference to a document that specifies specifies t specific the Tmodel's purpose.
3. Categories - Like all the other UDDI entities, tmode can be categorized.
4. Identifiers - The T-model can be associated with an arbitary no. of identifiers that uniquely ununiquely identify it.

UDDI entities are categorized through T-model via - taxonomies

Every T-Model ~~ho~~ T-model has a unique TModelKey for identification.

Scanned by CamScanner

Support

Categorization - also known as classification in JAXR terminology - is considered.

* Publishing API

The UDDI publish API allows you to publish, delete or update information contained in a UDDI registry.
~~Use UDDI publish API allows you to~~

The UDDI version 3 registry supports the following Publication API calls

1. add-publishedAssertions
Causes one or more publisherAssertion to be added to an individual publisher's assertion collection.

2. delete_binding
Cause one or more instance of bindingTemplate data to be deleted from the UDDI registry

3. delete_business
Removes one or more business registration & all direct contents from a UDDI registry.

4. delete_PublisherAssertions - Causes one or more PA element to be removed from a Publisher's assertion collection.

5. delete_Service
Removes one or more businessService element from the UDDI registry & from its containing businessEntity parent

6. delete_tModel - Logically deletes one or more tModel structure.

7. get_assertionStatusReport - Using this msg, a publisher can see the status of assertion that they have made as well as see the assertion that other have made.
It determines the status of current publisher assertion.

8. get_publisherAssertion - Obtain the full set of publisher assertions that are associated with an individual publisher account.
PublisherAssertion are used to control publicly visible business relationship.

9. get_registerInfo - Gets a list of all businessEntity & T-Model data.

10. Save_binding - Save or update a complete bindingTemplate element. Used to add or update one or more bindingTemplate elements.

Scanned by CamScanner

Save-business - Saves or update information about a complete business business Entity element.

Save-Service - Add or update information about a complete businessservice element.

Save tModel - Add or update one or more registered t-model elements.

Set-publisherAssertion - Manage all of the tracked relationship assertion associated with an individual publisher account.
PA is used to establish relationship between two <businessEntity> structures.

* Many business & organization are not effectively represented by a single businessEntity.
So soln is to use the PublisherAssertion structure. - Such a set of businessEntity structure represents a more or less coupled community.

Adding / Modifying Information	Deleting Information
① Save-business	① delete-business
② Save-services	② delete-Service
③ Save-binding	③ delete-binding
④ Save-tModel	④ delete-tModel
⑤ Save-PublisherAssertions	⑤ delete-PublishAssert
⑥ Set-PublisherAssertions	

Limitations of UDDI

① There are few universal UDDI registries in operation (maintained by IBM, microsoft, SAP etc).

② Most of the entries in them did not work or did not correspond to any real service.

③ UDDI supports for storing only the basic data structures, such as - business, users, service & service technical information.

④ UDDI is just a registry as opposed to ebXML Registry / Repository. which is the name suggest Registry as well as repositories. The difference is registry stores only metadata of object submitted & repository actually stores the submitted objects.

⑤ Has only a simple information Model.

Scanned by CamScanner

Support for Categorization in UDDI Regis

Categorization is also known as a classification in JAXR terminology - is considered to be the prominent functional of any registry.

Categorization enables the data to be classified with the help of various categorization systems (also known as taxonomies or classification schemes) such as - industry categorization system or a geography categorization system.

ex - A business can be classified as being located in the United States with the help of standard geography cate system such as - ISO 3166.

Categorizing data aids in searching for a particular piece of data. So cate. is useful in the discovery of information managed by a UDDI registry.

UDDI registry supports 3 industry standard categorization system which are built in.

The cate. system is represented by - <tModel> structure.

① UDDI - Supported Categorization System

The UDDI supported categorization system & their <tModel> names are-

① NAICS
② UNSPSC
③ ISO 3166
④ Operator Specific

① NAICS -

<tModel> Name - ntis-gov:naics:1997

This is a standard industry & service categorization system. NAICS stands for North American Industry Classification System.

② UNSPSC -

<tModel> name - unspsc-org:unspsc:3-1

UNSPSC is standard industry & service categorization system. UNSPSC stands for Universal Standard Products & Service classification. This was the first such industry classification schema defined for electronic business.

③ ISO 3166 -

<tModel> name - iso-ch:3166:1999

This is the standard geography-based cate. system.

④ Operator Specific

<tModel> name - uddi-org:general_keywords - This cate. system is operator specific. This is an open-ended cate. system that is not predefined.

② Checked Categorization System - Checked cate. system is used when the publisher of cate. system wishes to ensure that the categorization code values registered represent accurate & validate information.

The categorization code values represented by UDDI structure <categoryBag> would be checked by valid values during <save_business>, <save_service>, <save_tModel> API calls.

Scanned by CamScanner

③ Unchecked Categorization System.
Unchecked categorization system are used for categorization without need for a UDDI to perform validation of categorization code values.

Programming with UDDI
The API used for communicating with a UDDI Registry.
* The UDDI Programming API.
The UDDI specification defines two XML-based programming API for communicating with the UDDI registry node: Inquiry API & Publishing API.

I * Inquiry API
Inquiry API consist of XML msg defined using a UDDI schema, which can be used to locate information about a business. A UDDI programmer would use these inquiry APIs to retrieve information stored in the registry. To retrieve information a registry user does not need to be authenticated.
The following list of inquiry API function that can be used for finding information in a UDDI registry.
① <find_business> - The find_business API call returns a businessList msg.
② <find_relatedBusinesses> - returns a relatedBusinessList msg.
③ <find_service> - returns a serviceList msg.
④ <find_binding> - returns a bindingDetail msg.
⑤ <find_tModel> - returns a This <find_tModel> API call is for locating a list of tModel entries that match a set of specific criteria. This result will be returned in a tModelList msg.

To get further detailed information from the UDDI registry, the following inquiry API function are available.
<get_businessDetail> - returns complete businessEntity information.
<get_bindingDetail> - returns binding Template information.
<get_businessDetailExt> - returns extended bu businessEntity information.
<get_serviceDetail> - returns full information about a known business service.
<get_tModelDetail> - used to request full information about tModel data.

II Publishing API
Publishing API consist of function represented by a UDDI schema, which defines XML msg that can be used to create, update, delete information present in a UDDI registry.
Note - To publish to a UDDI registry, the registry user needs to be authenticated.

Scanned by CamScanner

Publishing Information to a UDDI Registry
Searching Information in a UDDI Registry
Deleting Information in a UDDI Registry

Publishing

- You can publish & manage information about your business, its organization, web service to a UDDI registry.
- After the business or service is published, the information is accessible to the clients of the registry.

1. You can publish – Business, Service,

Business - You can publish & mange inform about your business 1. You can list categories & identifier to which the business belongs, which provides additional way for client to search your business for perticular service.

Adding Business -

① Expand the Publish folder.
② Right click the Published Businesses folder & select Add Business.
③ Follow Add Business entity wizard to add a business.

The business properties has - Name, Description, Key, Related Business.
You can add – Service, contact, discovery URL, categories, identifiers.

Searching

You can search a registry by business, service or tModel entry.

Searching a registry.

1. Select the Search folder
2. Complete the Search option for the type of search you want to perform & click Search
3. When the Search completes, Click the Results folder to view the results
4. Click any of the items returned from the Search to view additional information about a business, service or tModel.

Search Type -

① Business

Options

* Business name - Enter a name of business for which you are searching.
* Sort by name - Select this check box & click Ascending or Descending.
* Sort by date – by date they were created or modified.
* Case Sensitive - Consider case when the performing search.
* Exact Match - exactly math the business name for searching business.
* Advance Option - by categories & identifiers

② Service -

Option.

* Service name
* Sort by name

Scanned by CamScanner

* Sort by date
* Case sensitive
* Exact Match
* Add Category

③ tModel

options

* tModelname
* Sort by name
* Sort by date
* Case sensitive
* Exact match
* Add category
* Add identifier

Deleting Information

The delete information operation allow you to remove information from the UDDI registry. The SOAP request & SOAP Response msgs used for the delete operations are

<delete - business>
<delete - Service>
<delete - binding>
<delete - tModel>
<delete - publisher Assertion>

are used to delete information from the UDDI Registry.

Scanned by CamScanner

5 Cloud Computing

Cloud computing - The cloud computing is a type of computing that relies on sharing computing resources rather than having local servers or personal device's to handle applications.

A cloud service has 3 distinct characteristics that differentiate it from traditional ~~traditional~~ hosting. - * It is sold on demand, typically by the minute or the hour.

* It is elastic - a user can have as much or as little of a service as they want ~~want~~ at any given time.

* The service is fully managed by the provider.

Cloud computing is a general term for anything that involves delivering hosted service over the internet. These services are broadly divided into 3 categories :- IaaS, PaaS, SaaS.

The name cloud computing was inspired by the cloud symbol that's often used to represent the internet in flowchart & diagram.

Characteristics

1. On-demand Self Service - A consumer can independently & unilaterally provision computing capabilities such as compute time, N/w connectivity & storage as needed automatically without requiring human interaction with each services provider.

2. Broad N/w Access - Capabilities are available over the N/w & accessed through standard mechanism that promote use by heterogeneous thin or thick client platform.

3. Resource Pooling - The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical & virtual resource dynamically assigned & reassigned according to consumer demand.

4. Rapid Elasticity - Capabilities can be rapidly & elastically provision, in some cases automatically to quickly scale out & rapidly redecised to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited & can be purchased in any quantity at any time.

5. Measured Service - Cloud system automatically ~~control~~ control & optimize resource use by leveraging a metering capabilities at some level of abstraction appropriate to the type of service (storage, compute, bandwidth, active user, accounts etc). Resource usage can be monitored, controlled controlled & reported, providing transparancy for both the provider & consumer of the utilized service.

Keep control over access.

Scanned by CamScanner

Service Models - (SaaS)

① Could S/w as a Service - The capability. The cloud application services or S/w as a Service (SaaS), represent the largest Cloud market and are still growing quickly.

SaaS uses the web to deliver appln that are managed by a third party vendor & whose interface is accessed on the client's side.

Most SaaS appln can be run directly from a web browser without any download or installation required.

Ex - Google Gmail, Salesforce, WebEx.

Common use of SaaS is replacing traditional on-device S/w.

SaaS is a cloud model that deliver on-demand appln that are hosted & managed by the service provider. SaaS is a way of delivering appln over the internet - as a service. SaaS appln are sometime called web-based S/w, on demand S/w, hosted S/w. SaaS appln runs on SaaS provider's server.

Cloud

② PaaS - Platform as a Service

Paas is a category of cloud computing service that provides a computing platform.

Paas offering facilitate the deployment of appln or services without the cost & complexity of buying & managing the underlying h/w / S/w.

Paas provides a platform & environment to allow developers to build appln & services over the internet.

Paas allows user to create S/w application using tools supplied by the provider.

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer created or acquired appln created using programming languages, libraries, services & tools supported by the provider.

Paas is outgrowth of SaaS, a S/w distribution model in which hosted S/w appln are made available to customers over the internet. Paas has several advantages for developer. With PaaS, operating system features can be changed & upgraded frequently.

Paas is a way to rent h/w, O.S, storage & n/w capacity over the internet. The service delivery model allows the customer to rent virtualized service. & they don't need to purchase h/w

Scanned by CamScanner

themselves. ~~or~~ This leaves them free to focus on the development of appln.

The client will only need to rent the resource. ~~the~~

Makes development possible for Nonexpert. With some PaaS offering anyone can develope an appln. They can simply do this through their web browser utilizing one-click functionality.

ex - Salient -, Google App Engine.

Security - Security is provided, including data security, backup & recovery.

A paas offering supplies an operating environment for the developing appln. In other word, it provides the architecture as well as the overall infrastructure to support appln developing. This includes n/wing, storage, s/w support & management services.

Cloud

③ IaaS - Iaas Infrastructure as a Service

Iaas is fundamental service model of cloud computing.

The capability provided to the consumer is to provision processing, storage, N/w & other fundamental computing resources where the consumer is able to deploy & run s/w, which can include O.S & appl^ns. The consumer does not manage or control the underlying cloud physical infrastructure but has control over O.S, storage, storage, deployed appln.

Iaas is defined as a computer infrastructure, such as virtualization, being delivered as a service. Iaas is popular in the data center where the s/w & servers are purchased as a fully outsourced service & usually billed on usage & how much of the resource is used.

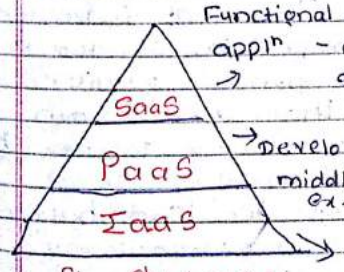


Fig- Cloud layers

→ Functional specific subscription based appln - on demand. ex - SalesForce.com, sendGrid.

→ Developement platforms, runtimes, middleware, Operating system. ex - Cloudbees, Microsoft Azure.

→ Pay by usage compute, memory, disk, N/w. ex - Amazon EC2, Rackspace, IBM softlayer.

EC2 - Elastic Computic Cloud

Scanned by CamScanner

Deployment Model -

① **Public Cloud** - ~~A public~~ Cloud ~~Servic~~
A public cloud sells services to anyone on the internet.
The cloud infrastructure is made available to the general public or a large industry group & is owned by an organization selling cloud services.
Ex - Amazon web Services.

② **Private Cloud** - A private cloud is a proprietary n/w or a data centre that supplies hosted services to a limited no. of people. It is managed by third party.

③ **Community Cloud** - The cloud infrastructure is shared by several organization & supports a specific community. It may be managed by organization, third party.

④ **Hybrid Cloud** - The cloud infrastructure is a ~~made~~ composition of two or more clouds (private, public or community) that they remain unique but are bounded together by standardized technology.
Ex - Cloud bursting for load balancing between clouds

Virtulization

In computing, virtulization means to create a virtual version of a device or resource such as server, storage device, n/w or even an operating system where the framework divides resource into one or more execution environments.
Virtulization s/w makes it possible to run multiple o.s & multiple appl^n on the same server at the same time.

Virtual Machine Technology - A VM (Virtual Machine) is an environment, usually a program or operating system, which does not exist physically exist but is created within another environment.

Advantages.

① **Maximize Resources** - Virtualization can reduce the no. of physical system you need to acquire & you can get more value out of the servers. Virtulization allows maximum use of h/w investment.

② **Multiple Systems** - With Virtulization, you ~~for~~ you can also run multiple types of application & even run different o.s for those appln on the same phy. h/w.

③ **IT Budget Integration** - When you use virtulization, management, administration & all the attendant requirement of managing your own infrastructure remain a direct cost of your IT operation.

Scanned by CamScanner

virtualisation platform runs on
↓
Host client/ server O.S.

VM are created within a Virtulization layer such as a hypervisor or a virtulization platform that runs on top of a client or server O.S. This O.S is known as host OS. Virtulization layer can be used to create many individual, isolated VM environment.

A VM is called a Guest while the environment its run within is called a host.

VM are often created to execute an instruction set different than that of the host environment

Virtual Machine Monitor (VMM)

A VMM is a host program that allows a single computer to support multiple, identical execution environment.

A VMM is a ~~sto~~ ~~softw~~ software program that enables the creation, management & governance of VM (virtual Machine) & mangl the operation of a virtualized environment.

A VMM is also known as - Virtual Machine Manager or Hypervisor.

VMM is the primary slw behind virtualization environment & impln. VMM facilites the creation of VMs, each with seperate O.S & appln. VMM manages the backend the backend operation of these VMs by allocating the necessary computing, memory, storage & other I/O resources

Hypervisor -

also called a Virtual machine manager, is a program that allows multiple O.S to share a single H/w processor.

Hypervisor is actually controlling the host preprocessor & resources, allocating what is needed to each O.S in turn & making sure that the guess O.S (VM) cannot disrupt each other.

In a virtulization technology, hypervisor is a slw program ~~th~~ that manages multiple O.S on single comp. system.

The hypervisor manages the system's processor, memory & other resources to allocates what each O.S requires.

Hypervisors are designed for a a particular processor architecture & may also be called virtualization managers.

Scanned by CamScanner

Type1

Xen hypervisor is distintctive open source technology, developed collaborativ by the Xen community.

It resides between the VMs & hardware as shown in the figure.

Type1 Hypervisor

Mainly there are two types of hypervisor i.e Type1 & Type2

Hypervisor is a firmware or low level program that acts as a Virtual machine manager.

Typ1 hypervisor runs on bare system.

ex - Oracle VM, RTS Hypervisior

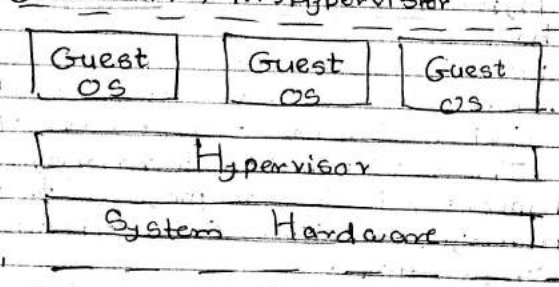


Fig - Typ1 Hypervisor

The Type1 hypervisor does not have any host O.S because they are installed on a bare system.

Bare System - is a computer system or N/w in which a virtual machine is installed directly on hardware rather than within the host o.s.

The bare metal refers to "H/w "harddisk.

Typ1 hypervisor also known as - native or bare - metal hypervisor.

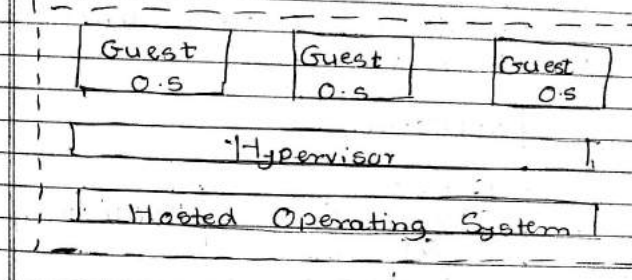
These run directly on the host computer's H/w to control the H/w resources & to manage guest o.s.

Type1 supports H/w virtualization.

Type 2 Hypervisor.

Type2 hypervisor runs as an applicatio on a host operating system.

ex - KVM, VMWare Fusion, VMWare Work Station



Type2 supports S/w virtubization.

Type 2, which operates as an appl^n on top of an existing O.S.

A Type1 hypervisor provides better performance & greater flexibility because it operates as a thin layer designed to expose H/w resources to VM.

Scanned by CamScanner

QO QEMU - Quick Emulator.

Type1 is more efficient than Type2 hypervisor.

KVM Hypervisor:

KVM hypervisor is the virtulization layer in Kernal-based Virtual machine.

In KVM, the Linux Kernel act as a Type 2 hypervisor.

The KVM is a type 2 hypervisor maintained by Qumranet.

KVM is based on the QEMU.

The main focus of KVM development is to use the X86 VT extension.

KVM is uses a set of linux kernel module to provide VT support.

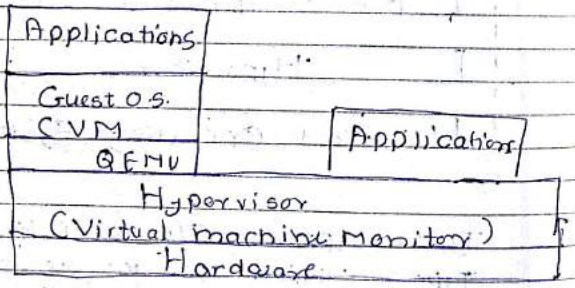


Fig - Virtulization component with KVM

Using KVM, one can run multipe virtual machines running unmodified Linux or Windows images.

KVM supports the QEMU Copy on Write (QCOW) disk image format.

Multiple VM can run on one disk image.

Xen Hypervisor -

A Xen Virtual environment consist of several items that work together to deliver the virtulization environment a customer is looking to deploy.

It contains -

1. Domain 0
2. Guest
3. Xen Hypervisor.
4. Physical hardware.

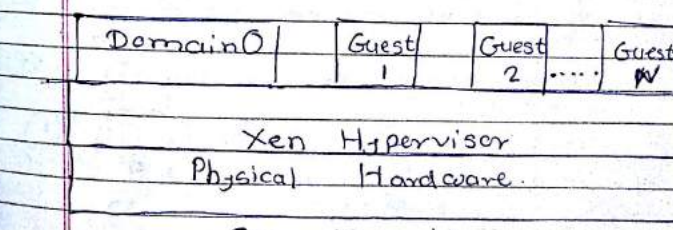


Fig - Xen Hypervisor

Xen hypervisor is the basic abstraction layer or s/w that sits directly on the H/w below any OS. It is responsible for CPU scheduling & memory partitioning of the various Virtual machines

Scanned by CamScanner

running on the H/w device.
The hypervisor not only abstract the H/w for the virtual machine but also controls the execution of VM's as they share common processing environment.

Domain 0

Domain 0, a modified Linux kernel, is a unique VM running on the Xen hypervisor that has special rights to access physical I/O resources as well as interact with the other VM.

All Xen virtulization environment require Domain 0 to be running before any other VM can be started.

VMWare Hypervisor

VMWare is one of the most popular solⁿ for data center virtulization. VMWare's hypervisor is available in two version –
the paid ESX full version & the free ESXi version.

The VMware ESX & VMware ESXi provides the foundation for building a reliable & dynamic IT infrastructure. VMWare ESX & ESXi are the most widely deployed hypervisor.

Benefits -

1. Decrease h/w, power & cooling cost by running multiple O.S on same physical servers.
2. Lower management overheads costs by reducing h/w.
3. Guarantee high level of performance.

Key Features -

1. Record-setting performance with up to 8,900 database transaction per second, 200,000 I/O operations per second & up to 16,000 exchange maiboxes on a single physical host.
2. Up to eight way virtual SMP (Symmetric multiprocessing) enabling the virtulization of multiprocessor workloads.

Scanned by CamScanner

(8) Broadest O.S supports of any hypervisor enabling IT to virtulize numerous version of windows, Linux.

(9) Built in high availability through N/C teaming & HBA multipathing to protect against H/W component failures.

(5) Up to 64 logical processing core, 256 virtual CPU, & 1TB RAM per host, enabling higher consolidation ratios.

1
2
3

Cloud Computing Architecture.

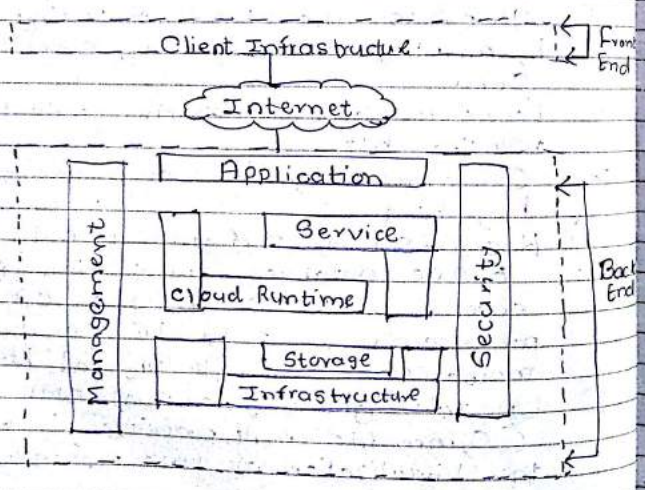


Fig - Cloud Computing Architecture

The cloud computing architecture comprises of many cloud components. Each of them are loosely coupled. We broadly divided the cloud architecture into two parts -

(1) FrontEnd
(2) BackEnd

Each of the ends are connected through a N/W usually via internet. The above diagram shows the graphical view of cloud computing architecture.

(1) Front End - The frontend refers to the client part of cloud computing system. It consist of interfaces & applications that are required to access access the cloud computing platforms ex. Web Browser.

(2) Back End - Back End refers to the cloud itself. It consist of all the resources required to provide cloud computing services. It comprises of huge data storage, virtual machines, security mechanism, services, deployment model, servers etc.

Important Points-

It is the responsibility of the back end to provide built-in security mechanism, traffic control & protocols.

The server employs certain protocols known as middleware helps the connected devices to communicate with each other.

Scanned by CamScanner

Multitenancy

A tenant is any application. Multitenant means multi tenants sharing resources. Multitenancy can be economical because s/w development & maintainance cost are shared. Multitenancy is a common attributes of both public & private clouds, and it applies to all three layers of a cloud - IaaS, PaaS, SaaS.

In cloud computing, the meaning of multi-tenancy architecture has broadened because of new service models that take advantage of virtualization & remote access. A s/w-as-a service (SaaS) provider, for example, can run one instance of its application on one instance of a database & provide web access to multiple customer.

Multitenancy is a reference to the mode of operation of s/w where multiple independent instance of one or multiple applications operate in shared environment. The tenants are logically isolated but physically integrated.

Multitenant refers to a principle in a s/w architecture where a single instance of the s/w runs on a server, serving multiple tenants. In such environment, multiple customer share the same application, running on the same OS on the same H/w, with same data storage mechanism.

Scanned by CamScanner

Cloud - storing & accessing data & programs over the internet.

-x-

Web Services

Web services are small units of code designed to handle a limited set of tasks

e.g prog. designed to supply other appl'ns with the latest stock exchange

- Web services are application components
- Web services communicate using open protocols
- Web services are self-contained and self-describing
- Web services can be discovered using UDDI
- Web services can be used by other applications
- HTTP and XML is the basis for Web services

Web services use the standard web protocols HTTP, XML, SOAP, WSDL & UDDI

Interoperability has Highest Priority

When all major platforms could access the Web using Web browsers, different platforms couldn't interact. For these platforms to work together, Web-applications were developed.

Web-applications are simply applications that run on the web. These are built around the Web browser standards and can be used by any browser on any platform.

Web Services take Web-applications to the Next Level

By using Web services, your application can publish its function or message to the rest of the world.

Web services use XML to code and to decode data, and SOAP to transport it (using open protocols).

With Web services, your accounting department's Win 2k server's billing system can connect with your IT supplier's UNIX server.

Web Services have Two Types of Uses

Reusable application-components.

There are things applications need very often. So why make these over and over again?

Web services can offer application-components like: currency conversion, weather reports, or even language translation as services.

Connect existing software.

Scanned by CamScanner

Web Services

- Web services are application components
- Web services communicate using open protocols
- Web services are self-contained and self-describing
- Web services can be discovered using UDDI
- Web services can be used by other applications
- HTTP and XML is the basis for Web services

Interoperability has Highest Priority

When all major platforms could access the Web using Web browsers, different platforms couldn't interact. For these platforms to work together, Web-applications were developed.

Web-applications are simply applications that run on the web. These are built around the Web browser standards and can be used by any browser on any platform.

Web Services take Web-applications to the Next Level

By using Web services, your application can publish its function or message to the rest of the world.

Web services use XML to code and to decode data, and SOAP to transport it (using open protocols).

With Web services, your accounting department's Win 2k server's billing system can connect with your IT supplier's UNIX server.

Web Services have Two Types of Uses

Reusable application-components.

There are things applications need very often. So why make these over and over again?

Web services can offer application-components like: currency conversion, weather reports, or even language translation as services.

Connect existing software.

Scanned by CamScanner

Web services can help to solve the interoperability problem by giving different applications a way to link their data.

With Web services you can exchange data between different applications and different platforms.

.....Different books and different organizations provide different definitions to Web Services. Some of them are listed here.

- 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.
- Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

To summarize, 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

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)

Scanned by CamScanner

All these components have been discussed in the Web Services Architecture chapter.

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.

You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.

You can also use C# to build new web services on Windows that can be invoked from your web application that is based on JavaServer Pages (JSP) and runs on Linux.

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.

* Benefits.....Here are the benefits of using Web Services:

Exposing the Existing Function on the network

A web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. Web services allows you to expose the functionality of your

Scanned by CamScanner

existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.

Interoperability

Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use the web services. For example, a VB or .NET application can talk to Java web services and vice versa. Web services are used to make the application platform and technology independent.

Standardized Protocol

Web services use standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack. 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. This solution is much less costly compared to proprietary solutions like EDI/B2B. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.

characteristics :-

Web services have the following special behavioral characteristics:

XML-Based

Web Services uses XML at data representation and data transportation layers. Using XML eliminates any networking, operating system, or platform binding. Web Services based applications are highly interoperable application at their core level.

Loosely Coupled

A consumer of a web service is not tied to that web service directly. The web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

Scanned by CamScanner

platform independent
support document exchange.

Coarse-Grained

Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service.

Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.

Ability to be Synchronous or Asynchronous

Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions.

Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

Supports Remote Procedure Calls(RPCs)

Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.

Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms.

A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

Supports Document Exchange

One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be as simple as representing a current address, or they can be as complex as representing an entire book or Request for Quotation (RFQ). Web services support the transparent exchange of documents to facilitate business integration.

Web services are used to transfer simple to complex documents.

Scanned by CamScanner

.....There are two ways to view the web service architecture:

- The first is to examine the individual roles of each web service actor.
- The second is to examine the emerging web service protocol stack.

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.

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.

Scanned by CamScanner

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.

The next chapter explains the components of web services.

Few Words about Service Transport

The bottom of the web service protocol stack is service transport. This layer is responsible for actually transporting XML messages between two computers.

Hyper Text Transfer Protocol (HTTP)

Currently, HTTP is the most popular option for service transport. HTTP is simple, stable, and widely deployed. Furthermore, most firewalls allow HTTP traffic. This allows XML-RPC or SOAP messages to masquerade as HTTP messages. This is good if you want to integrate remote applications, but it does raise a number of security concerns.

Blocks Extensible Exchange Protocol (BEEP)

This is a promising alternative to HTTP. BEEP is a new Internet Engineering Task Force (IETF) framework for building new protocols. BEEP is layered directly on TCP and includes a number of built-in features, including an initial handshake protocol, authentication, security, and error handling. Using BEEP, one can create new protocols for a variety of applications, including instant messaging, file transfer, content syndication, and network management.

SOAP is not tied to any specific transport protocol. In fact, you can use SOAP via HTTP, SMTP, or FTP. One promising idea is therefore to use SOAP over BEEP.

.....The term *Web services* describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used primarily as a means for businesses to communicate with each other and with clients, Web

Scanned by CamScanner

services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall.

Unlike traditional client/server models, such as a Web server/Web page system, Web services do not provide the user with a GUI. Web services instead share business logic, data and processes through a programmatic interface across a network. The applications interface, not the users. Developers can then add the Web service to a GUI (such as a Web page or an executable program) to offer specific functionality to users.

Web services allow different applications from different sources to communicate with each other without time-consuming custom coding, and because all communication is in XML, Web services are not tied to any one operating system or programming language. For example, Java can talk with Perl, Windows applications can talk with UNIX applications.

.....

.....A **Web service** is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the Web with the service *always on* as in the concept of utility computing. The W3C defines a Web service generally as:

a software system designed to support interoperable machine-to-machine interaction over a network.[1]

The W3C Web Services Architecture Working Group defined a Web Services Architecture, requiring a specific implementation of a "Web service." In this:

[a Web service] has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.[1]

The W3C also states:

We can identify two major classes of Web services:

- *REST-compliant Web services*, in which the primary purpose of the service is to manipulate representations of Web resources using a uniform set of stateless operations.
- *Arbitrary Web services*, in which the service may expose an arbitrary set of operations.[2]

Contents

Scanned by CamScanner

information systems. J2EE also provides excellent client connectivity capabilities, ranging from PDA to Web browsers to Rich Clients (Applets, CORBA applications, and Standard Java Applications).

Figure 1.7 shows various components of the J2EE architecture.

A typical J2EE architecture is physically divided in to three logical tiers, which enables clear separation of the various application components with defined roles and responsibilities. The following is a breakdown of functionalities of those logical tiers:

Presentation tier.

The Presentation tier is composed of Web compo-

Scanned by CamScanner

nents, which handle HTTP requests/responses, Session management, Device independent content delivery, and the invocation of business tier components.

Figure 1.7
J2EE application architecture.

IIOP
Applets/
Applications
J2EE Server
WEB CONTAINER
EJB CONTAINER
LEGACY
APPLICATIONS
HTTP
Web Browser PDA
HTTP
ii
CLIENTS
PRESENTATION
TIER
APPLICATION
TIER
INTEGRATION
TIER
DATABASE
SQL/JDBC

03236403 Ch01.F 11/14/02 9:26 AM Page 18

Application tier.

The Application tier (also known as the Business tier) deals with the core business logic processing, which may typically deal with workflow and automation. The business components retrieve data from the information systems with well-defined APIs provided by the application server.

Integration tier.

The Integration tier deals with connecting and communicating to back-end Enterprise Information Systems (EIS), database applications and legacy applications, or mainframe applications. With its key functionalities and provisions for partitioning applications into logical tiers, J2EE has been highly adopted as the standard solution for developing and deploying mission critical Web-based applications. The power of J2EE-based applications would be tremendous, if it is enabled to interoperate with other potential J2EE-deployed applications. This enables business components across networks to negotiate among them

Scanned by CamScanner

and interact without human interaction. It also enables the realization of syndication and collaboration of business processes across the Internet by enabling them to share data and component-level processes in real time. This phenomenon is commonly referred to as business-to-business (B2B) communication.

The emergence of the Extensible Markup Language (XML) for defining portable data in a structured and self-describing format is embraced by the industry as a communication medium for electronic data exchange. Using XML as a data exchange mechanism between applications promotes interoperability between applications and also enhances the scalability of the underlying applications. Combining the potential of a J2EE platform and XML offers a standard framework for B2B and inter-application communication across networks.

With J2EE enabling enterprise applications to the Internet and XML acting as a "glue" bridges these discrete J2EE-based applications by facilitating them to interoperate with each other. XML, with its incredible flexibility, also has been widely adopted and accepted as a standard by major vendors in the IT industry, including Sun, IBM, Microsoft, Oracle, and HP. The combination of these technologies offers more promising possibilities in the technology sector for providing a new way of application-to-application communication on the Internet. It also promotes a new form of the distributed computing technology solution referred to as Web services.

03236403 Ch01.F 11/14/02 9:26 AM Page 19

The Emergence of Web Services

Today, the adoption of the Internet and enabling Internet-based applications has created a world of discrete business applications, which co-exist in the same technology space but without interacting with each other. The increasing demands of the industry for enabling B2B, application-to-application (A2A), and inter-process application communication has led to a growing requirement for service-oriented architectures. Enabling service-oriented applications facilitates the exposure of business applications as service components enable business applications from other organizations to link with these services for application interaction and data sharing without human intervention. By leveraging this architecture, it also enables interoperability between business applications and processes. By adopting Web technologies, the service-oriented architecture model facilitates the delivery of services over the Internet by leveraging standard technologies such as XML. It uses platform-neutral standards by exposing the underlying application components and making them available to any

Scanned by CamScanner

application, any platform, or any device, and at any location. Today, this phenomenon is well adopted for implementation and is commonly referred to as Web services. Although this technique enables communication between applications with the addition of service activation technologies and open technology standards, it can be leveraged to publish the services in a register of yellow pages available on the Internet. This will further redefine and transform the way businesses communicate over the Internet. This promising new technology sets the strategic vision of the next generation of virtual business models and the unlimited potential for organizations doing business collaboration and business process management over the Internet.

Summary

In this chapter, we discussed the evolution and the basics of distributed computing technologies and the emergence of Web services that define the next generation of business services models and business process communication over the Internet.

In particular, we looked at the background of distributed computing; the fundamentals of distributed computing techniques; the basics of industry-accepted technologies like CORBA, RMI, DCOM, and MOM; the role of J2EE and XML for enabling distributed computing over the Internet; and the concept of service-oriented architectures and the emergence of Web services.

In the following chapters, we will go into a more detailed introduction to Web services concepts and focus on the various aspects of designing and developing Web services.

03236403 Ch01.F 11/14/02 9:26 AM Page 20

Today, people use the Internet as an everyday service provider for reading headline news, obtaining stock quotes, getting weather reports, shopping online, and paying bills, and also for obtaining a variety of information from different sources. These Web-enabled applications are built using different software applications to generate HTML, and their access is limited only through an Internet browser or by using an application-specific client. This is partially due to the limitations of HTML and the Web server-based technologies, which are primarily focused on presentation and their inability to interact with another application.

The emergence of Web services introduces a new paradigm for enabling the exchange of information across the Internet based on open Internet standards and technologies. Using industry standards, Web services encapsulate applications and publish them as services. These services deliver XML-based data on the wire and expose it for use on the Internet, which can be dynamically located, subscribed, and accessed using a wide

Scanned by CamScanner

range of computing platforms, handheld devices, appliances, and so on. Due to the flexibility of using open standards and protocols, it also facilitates Enterprise Application Integration (EAI), business-to-business (B2B) integration, and application-to-application (A2A) communication across the Internet and corporate intranet. In organizations with heterogeneous applications and distributed application architectures, the introduction of

Introduction to Web Services

CHAPTER

2

04236403 Ch02.F 11/14/02 9:26 AM Page 21

Web services standardizes the communication mechanism and enables interoperability of applications based on different programming languages residing on different platforms.

This chapter presents an introduction on Web services, especially focusing on the following:

- The definition of Web services
- Motivation and characteristics of Web services
- Web services industry standards and technologies
- Web services strategies and solutions
- Benefits of Web services

Today's leading technology vendors have set their strategies around providing infrastructure solutions for delivering Web services. With the increasing adoption, acceptance, and availability of platforms, languages, application tools, and supporting technology solutions, it is expected that Web services will become a new service industry providing businesses services over the Internet.

What Are Web Services?

Web services are based on the concept of service-oriented architecture (SOA). SOA is the latest evolution of distributed computing, which enables software components, including application functions, objects, and processes from different systems, to be exposed as services.

According to Gartner research (June 15, 2001), "Web services are loosely coupled software components delivered over Internet standard

Scanned by CamScanner

technologies."

In short, Web services are self-describing and modular business applications that expose the business logic as services over the Internet through programmable interfaces and using Internet protocols for the purpose of providing ways to find, subscribe, and invoke those services.

Based on XML standards, Web services can be developed as loosely coupled application components using any programming language, any protocol, or any platform. This facilitates delivering business applications as a service accessible to anyone, anytime, at any location, and using any platform.

Consider the simple example shown in Figure 2.1 where a travel reservation services provider exposes its business applications as Web services supporting a variety of customers and application clients. These business applications are provided by different travel organizations residing at different networks and geographical locations.

04236403 Ch02.F 11/14/02 9:26 AM Page 22

Figure 2.1

An example scenario of Web services.

The following is a typical scenario:

1. The Travel service provider deploys its Web services by exposing the business applications obtained from different travel businesses like airlines, car-rental, hotel accommodation, credit card payment, and so forth.Wireless

04236403 Ch02.F 11/14/02 9:26 AM Page 23

2. The service provider registers its business services with descriptions using a public or private registry. The registry stores the information about the services exposed by the service provider.
3. The customer discovers the Web services using a search engine or by locating it directly from the registry and then invokes the Web services for performing travel reservations and other functions over the Internet using any platform or device.
4. In the case of large-scale organizations, the business applications consume these Web services for providing travel services to their own employees through the corporate intranet.

The previous example provides a simple scenario of how an organization's business functionalities can be exposed as Web services and invoked by its customers using a wide range of application clients.

As we discussed earlier, Web services are typically implemented based on open standards and technologies specifically leveraging XML. The XML-based standards and technologies, such as Simple Object Access Pro-

Scanned by CamScanner

tocol (SOAP); Universal Description, Discovery, and Integration (UDDI); Web Services Definition Language (WSDL); and Electronic Business XML (ebXML), are commonly used as building blocks for Web services. These technologies are discussed briefly in the section
Core Web Services Stan-
dards
, which follows later.

Motivation and Characteristics

Scanned by CamScanner

System s/w - is computer s/w designed to operate & control the computer h/w & to provide a platform for running appln s/w.

Web Service vs Web Application

An application that the users access over the internet is called a web application. Generally, any software that is accessed through a web browser could be called a web application. Web applications have gained an immense popularity due to the easiness of use. According to W3C (World Wide Web Consortium) a web service is a system of software that allows different machines to interact with each other through a network. Web services achieve this task using XML, SOAP, WSDL and UDDI open standards.

What is a Web Application?

An application that the users access over the internet is called a web application. Generally, any software that is accessed through a web browser could be called a web application. When using web applications, users do not have to worry about installing and maintaining the application. Furthermore, web applications provide support for different platforms. Also, it is very easy to use web applications since the only requirement is a web browser. Due to these reasons, web applications have gained an immense popularity. Popular web applications include web mail applications, online auctions, wikis, etc. Generally, web applications are organized in to tiers, where each tier is responsible for a specific task. In the early days, web applications were made up of a single tier, while today, most web applications are built on the three-tier architecture and some complex applications use n-tier architecture (n>3). In the three-tier architecture, the three tiers are dedicated for presentation, application (or logic) and storage from top tier to bottom tier.

What is a Web Service?

A web service is a system of software that allows different machines to interact with each other through a network. Web services use XML, SOAP, WSDL and UDDI open standards to achieve this task. XML is a language that can be used to send messages with different platforms and different programming languages and it is used to tag data in web services. SOAP is a protocol that is based on XML that would allow applications to communicate over HTTP and it is used for accessing a web service. WSDL is used to describe and locate a web service. Web services are mainly used to achieve reusability of application components. Application components such as weather reports, currency converters, etc. are widely used. Therefore, without developing them again and again, they are offered as web services, which can be easily used. Welservices can also be used to exchange data between different applications running on different platforms.

What is the difference between a Web Application and a Web Service?

A web application is an application that is accessed through a web browser running on client's machine whereas a web service is a system of software that allows different machines to interact with each other through a network. Most of the times, web services do not necessarily have a user interface since it is used as a component in an application, while a web application is a complete application with a GUI. Furthermore, web services can be used to communicate or transfer data between web applications running on different platforms.

Scanned by CamScanner

he difference between the two might be fuzzy at the edges, but the basic idea is:

Web application: any application which resides on a server, but meant for use by humans, which uses web pages as the presentation layer. All user interactivity (the GUI) is done through web pages, but all data is stored and (mostly) manipulated on the server.

Web service: server-based application (as above) which may be accessed over the web via HTTP, but is meant primarily for interaction with other programs. Thus, it will have a clearly-defined API which consists of providing responses to HTTP GET and POST requests made by a remote application. Now, this doesn't mean you can't access a web service from your browser, but it means that the application won't **necessarily** have a GUI user interface. You will most likely, for example, receive all results of GET and POST requests as strings of XML, which requires a client-side parser.

So, think of web applications as completed user interfaces, while web services are more intended to be application components or "libraries" which can be used by other applications. Essentially, web services are to distributed applications what DLLs or class libraries are to most traditional compiled applications.

Also, this means that a web service has to have a more organized design that most web applications, because there won't necessarily be a human at the other end who can figure out through trial and error what the app does. This is why the whole concept of "web services" has brought a whole new set of acronyms such as SOAP, UDDI, etc...

(pauses for breath)

And furthermore--

Scanned by CamScanner

Blocks Extensible Exchange Protocol (BEEP)

This is a promising alternative to HTTP. BEEP is a new Internet Engineering Task Force (IETF) framework for building new protocols. BEEP is layered directly on TCP and includes a number of built-in features, including an initial handshake protocol, authentication, security, and error handling. Using BEEP, one can create new protocols for a variety of applications, including instant messaging, file transfer, content syndication, and network management.

SOAP is not tied to any specific transport protocol. In fact, you can use SOAP via HTTP, SMTP, or FTP. One promising idea is therefore to use SOAP over BEEP.

Web Services - Components

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.

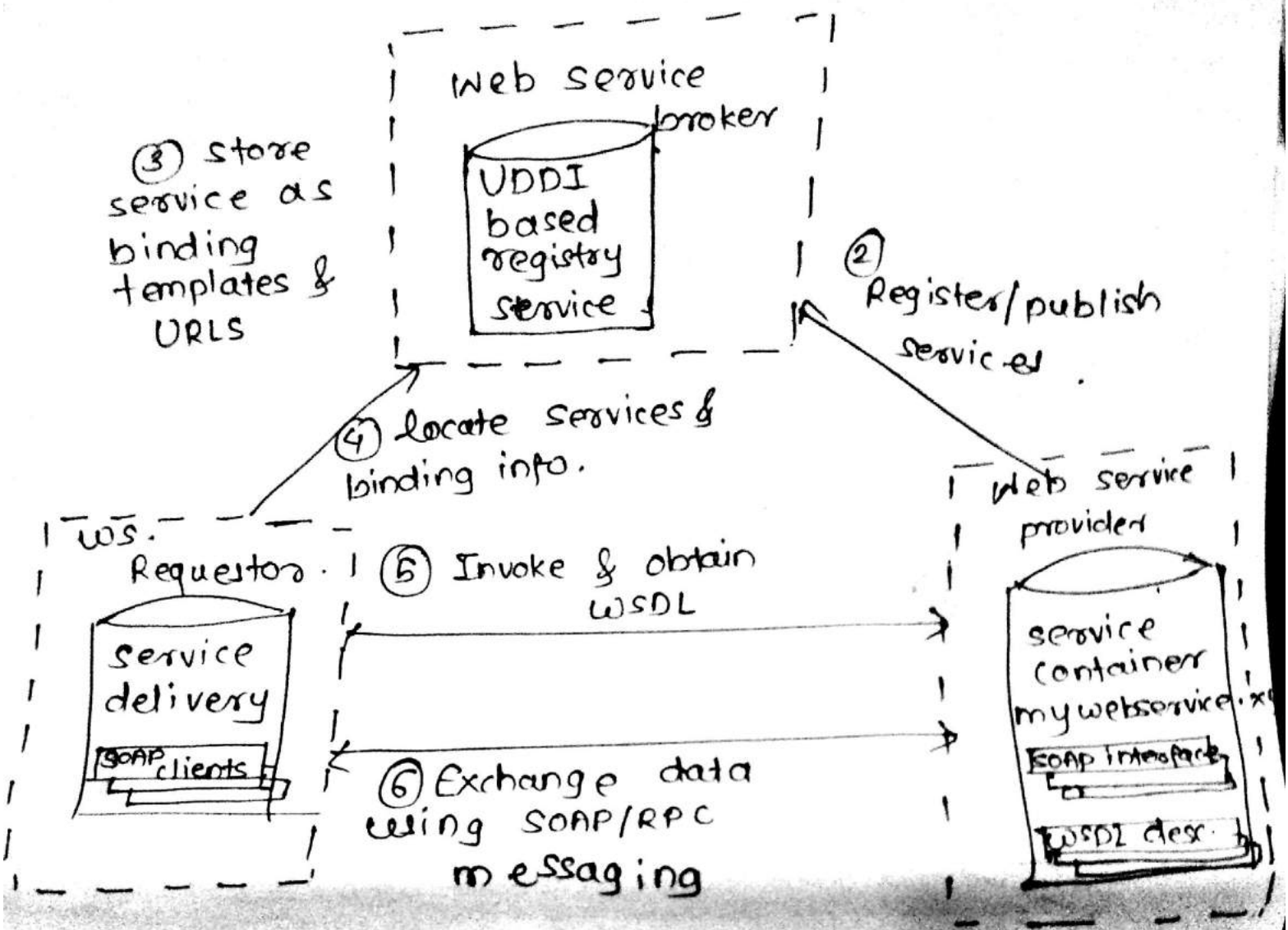
To learn more about XML-RPC, visit our XML-RPC Tutorial .

SOAP

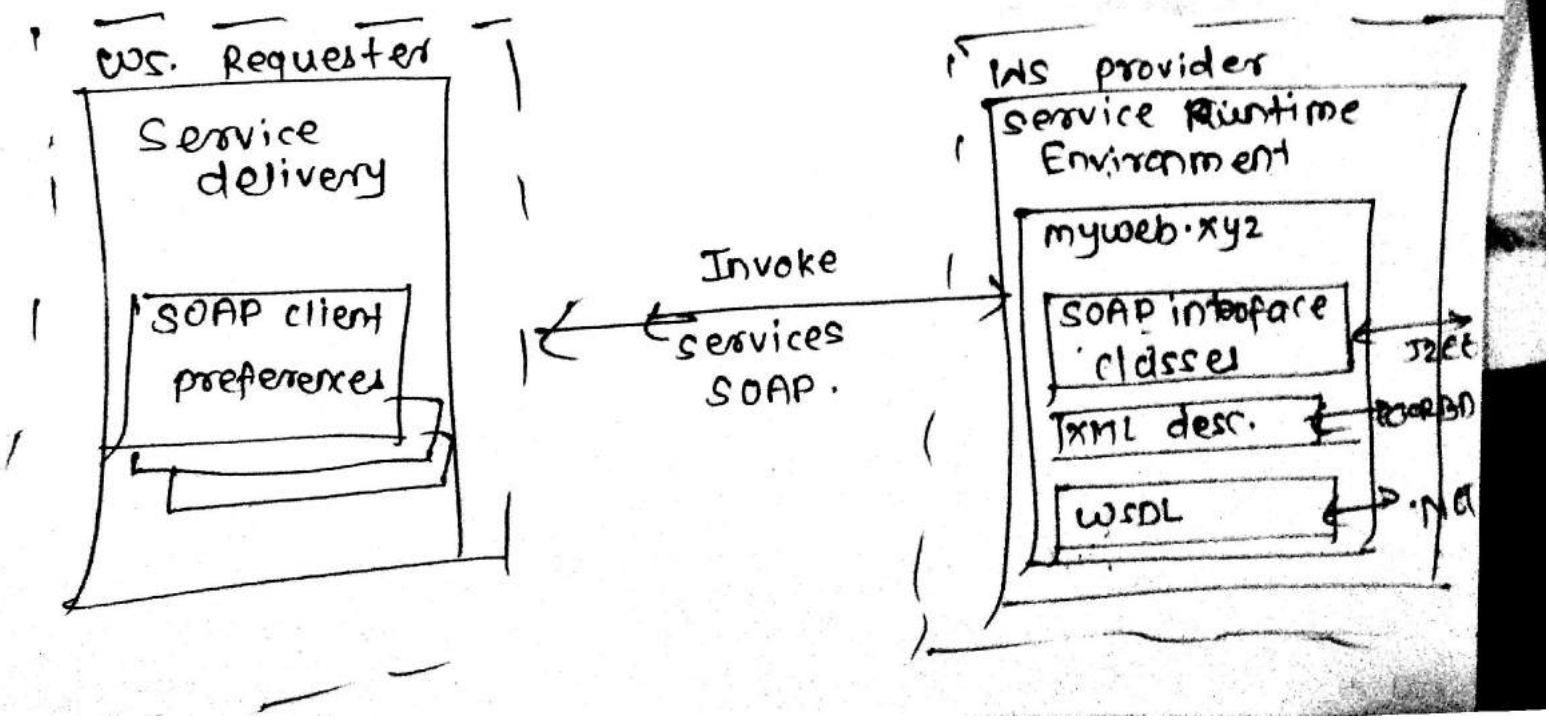
SOAP is an XML-based protocol for exchanging information between computers.

- 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.

Scanned by CamScanner



—x—



Scanned by CamScanner

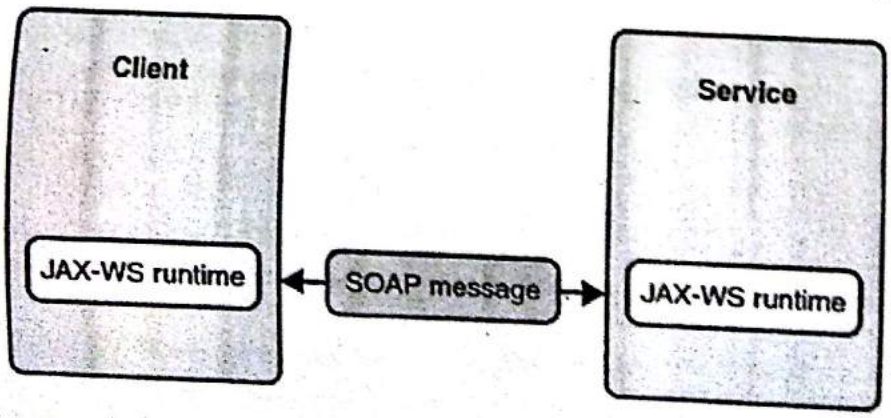
```
import javax.jws.WebService;
import javax.jws.WebMethod;

@WebService
public class AddNumbersImpl {
  @WebMethod(action="addNumbers")
  public int addNumbers(int number1, int number2)
      throws AddNumbersException {
    if (number1 < 0 || number2 < 0) {
      throw new AddNumbersException(
          "Negative number cannot be added!",
          "Numbers: " + number1 + ", " + number2);
    }
    return number1 + number2;
  }
```

Creating a Simple Web Service and Client with JAX-WS

This section shows how to build and deploy a simple web service and client. The source code for the service is in *tut-install*`/javaeetutorial5/examples/jaxws/helloservice/` and the client is in *tut-install*`/javaeetutorial5/examples/jaxws/simpleclient/`.

Figure 16-1 illustrates how JAX-WS technology manages communication between a web service and client.

Figure 16-1 Communication between a JAX-WS Web Service and a Client



The starting point for developing a JAX-WS web service is a Java class annotated with the `javax.jws.WebService` annotation. The `@WebService` annotation defines the class as a web service endpoint.

A **service endpoint interface** or **service endpoint implementation** (SEI) is a Java interface or class, respectively, that declares the methods that a client can invoke on the service. An interface is not required when building a JAX-WS endpoint. The web service implementation class implicitly defines an SEI.

Scanned by CamScanner

- The implementing class must not be declared `final` and must not be `abstract`.
- The implementing class must have a default public constructor.
- The implementing class must not define the `finalize` method.
- The implementing class may use the `javax.annotation.PostConstruct` or `javax.annotation.PreDestroy` annotations on its methods for life cycle event callbacks.

The `@PostConstruct` method is called by the container before the implementing class begins responding to web service clients.

The `@PreDestroy` method is called by the container before the endpoint is removed from operation.

Coding the Service Endpoint Implementation Class

In this example, the implementation class, `Hello`, is annotated as a web service endpoint using the `@WebService` annotation. `Hello` declares a single method named `sayHello`, annotated with the `@WebMethod` annotation. `@WebMethod` exposes the annotated method to web service clients. `sayHello` returns a greeting to the client, using the name passed to `sayHello` to compose the greeting. The implementation class also must define a default, public, no-argument constructor.

```
package helloservice.endpoint;

import javax.jws.WebService;

@WebService
public class Hello {
    private String message = new String("Hello, ");
public void Hello() {}

    @WebMethod
    public String sayHello(String name) {
        return message + name + ".";
    }
}
```

Building, Packaging, and Deploying the Service

You can build, package, and deploy the `helloservice` application using either NetBeans IDE or `ant`.

Building, Packaging, and Deploying the Service Using NetBeans IDE

Follow these instructions to build, package, and deploy the `helloservice` example to your Application Server instance using NetBeans IDE.

1. In NetBeans IDE, select File→Open Project.
2. In the Open Project dialog, navigate to *tut-install*`/javaeetutorial5/examples/jaxws/`.

Scanned by CamScanner

2 Web Services

2.1 Web Services definition

Different books and organizations give several complementary definitions of Web Services. Some of them are given here:

"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. Because 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" [Cer02].

"Web Services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or Web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML" [IBM09].

"A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards" [Wha09].

Thus, web services are platform-independent, based on XML messages. The idea is to distribute services over the Internet and to make them available for clients. These services can be implemented with any language and can be invoked as well as composed.

2.2 Web Services Architecture

There are three major roles within the web services architecture:

- Service provider
  This is the provider of the web service. The service provider builds the service and makes it available on the Internet for consumers.

- Service requestor
  This is any consumer of the web service. The requestor invokes an existing web service by opening a network connection and sending an XML-SOAP (see Subsection 3.1) request.

Scanned by CamScanner

- Service registry
  It is a centralized directory of services. The registry is used as a central place where providers or developers can publish new services or find existing ones. It therefore serves as a centralized clearinghouse for companies and their services.

Figure 2.1 gives a logical view of web services by illustrating the relationship between the web services roles and operations. First, the web service provider publishes its web services with the discovery agency. Next, the web service consumer looks for desired web services using the registry of the discovery agency. Finally, the web services client invokes the web services by using the information obtained from the discovery agency.

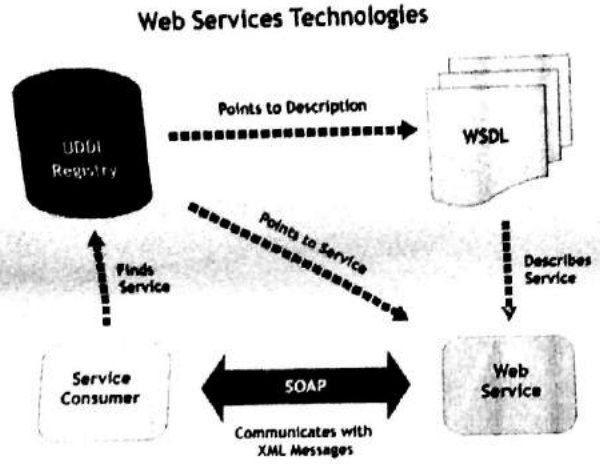


Figure 2.1 (taken from [Dmr02])

2.2.1 The Web Service Protocol Stack

The architecture of a Web Services stack varies from one organization to another. The number and complexity of layers for the stack depend on the organization. Web services are built by using various related technologies. Figure 2.2 illustrates the stack of specific, complementary standards on which web services are generally based on.

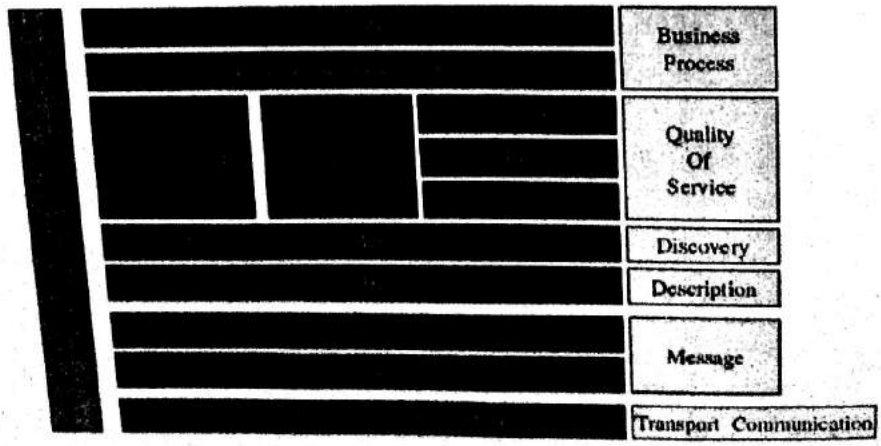


Figure 2.2 The Web Services technology stack (inspired from [Pap08])

Scanned by CamScanner

- Service transport
  The service transport layer delivers messages between applications. This layer usually implements hypertext transfer protocol (HTTP) [NWG99], Simple Mail Transfer Protocol (SMTP) [Wik092], file transfer protocol (FTP), and newer protocols, such as Blocks Extensible Exchange Protocol (BEEP) [Wha09].

- XML messaging
  This layer is responsible for encoding messages in a common XML [W3C08] format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP [Wha09].

- Simple object access protocol (SOAP)
  SOAP is a simple XML-based messaging protocol responsible for transferring data between different web services. It is built using XML and relies on common Internet transport protocol like HTTP to transport its messages. SOAP allows communication among interacting web services by implementing a request/response model and using HTTP to access networks protected by firewalls, which do not currently prevent HTTP and FTP service requests [Pap08]. See Section 3.1 for further information.

- Service description WSDL
  The purpose of this layer is to define the public interface of a specific web service. Currently, service description is realised through the Web Service Description Language (WSDL) which is based on XML [Wha09]. See Section 3.2 for further details about WSDL.

- Service discovery
  The service discovery layer registers services into a common repository and provides an easy publish/find mechanism. This layer is often implemented via Universal Description, Discovery, and Integration (UDDI) [Wha09]. But, the problem of service discovery is much discussed and the UDDI standard seems not to be used in large scale deployments. See Section 3.3 for more details.

- Service orchestration
  The topmost service orchestration layer is in charge of the execution logic of web services based applications by determining their control flows (e.g. conditional, sequential, parallel and exceptional execution). This layer enables enterprises to define and realise complex business processes [Pap08].

With the ongoing evolution of web services, it is possible that more layers will be added to these different layers, for example specifying quality of services (QoS) aspects, may be added to the technology stack described above as well.

2.2.2 Additional specifications, WS*

Several Web services specifications have recently been developed or are still being developed in order to enlarge Web Services functionalities. These specifications are generally referred to as WS-*. Below, some of these are briefly described.

- WS-Security
  The WS-Security specification determines the use of XML Encryption and XML

Scanned by CamScanner

Signature in SOAP so as to secure communication. It is used either as an alternative or an extension to using HTTPS to secure the message exchanges. As mentioned in [Erl06], the principal aspects of security addressed by these specifications are identification, authentication, authorization, integrity, confidentiality and non-repudiation.

- WS-Reliability
  The purpose of the WS-Reliability specification is to provide an environment that insures the delivery of a SOAP message or the reporting of a failure condition [Erl06].

- WS-Transaction
  The Web Services Transactions specifications enable Web services domains to interoperate and allow composing transactional qualities of service into Web services applications. These specifications describe an extensible coordination framework and specific coordination types for: Short duration, ACID transactions (WS-AtomicTransaction) and Longer running business transactions (WS-BusinessActivity). The interested reader is referred to [IBM04] for further details.

- WS-addressing
  The WS-addressing specification implements addressing extensions which are based on endpoint references and message information headers. Endpoint references allow the identification in a standardized way of a specific instance of a web service. Message information headers provide message exchange properties to a specific message, conveying interaction semantics to recipient services [Erl06].

- WS-policies
  The WS-Policies specification serves to attach features/characteristics (e. g. rules, behaviours, requirements and preferences) to Web resources, principally web services [Erl06].

- WS-Metadata exchange
  WS-Metadata exchange specification enables service requestors to issue requests to obtain metadata (such as WSDL, Schema, Policy Meta information) from services providers. This specification helps to improve the service description discovery process [Erl06].

- WS-Notification and eventing
  WS-Notification is a group of specifications in the context of the WS-Resource environment. These specifications allow event driven programming between web services by providing a protocol for web services to subscribe to another web service, or to accept a subscription from another web service [Wik09].

2.3 How do web services work?

The programmer builds a web service using a specific programming language. This service is published using a WSDL interface. This service can be invoked by a consumer "client" using this interface. Web services are presented to clients as a set of operations that provide business logic on behalf of the provider. Web services

Scanned by CamScanner

must be deployed on a server container to be available for consumers. On the client side, a remote object that represents the remote service must be generated. This allows clients to invoke the operations defined on the server side. The developer has not to care about creating or parsing SOAP messages. That task is performed by the web service's APIs runtime system. .Net web services for example can be invoked by any web service client and vice versa.

A Java-based Web Service built and deployed on Solaris operating system can be accessed from Visual Basic program which runs on Windows. Any language can also be used to realise new web services. These web services are invoked from any web application which is implemented using any other programming language and runs on any operating system.

An Example

Let us consider a simple application to sell cars. The buyer uses a client application built with Visual Basic or JSP to get information about a specific vehicle.

The processing logic for this application is written in Java and resides on a Linux machine which also interacts with a database to store the information.

The steps illustrated above are:

- *The client calls the web service method through a remote object which represents that web service.*
- The client program bundles the car information request 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 details concerning the selected car.
- Next, the Web Service packages up 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 car details process (the return result of the called method).

2.4 Web services characteristics

From [Pap08] and [Wha09] one can derive the following characteristics of web services:

- XML-based
  Web Services rely on XML for data representation and transportation. The use of XML avoids any network, operating system or platform binding.
- Loose coupling
  There is no direct tie between a web service and its user. Alterations of the WS

Scanned by CamScanner

interface do not deteriorate the user's capability of interacting with the service. Whereas in a tightly coupled system, the client and server logic are closely bound to each other, implementing a loosely coupled architecture facilitates software system management and helps the integration of different systems.

- Ability to be synchronous or asynchronous
  Ties between the client and the execution of the service are referred to as synchronicity. In synchronous invocations, the client sends his request and then waits for the response without being able to execute other operations during this period. In contrast, asynchronous invocations allow clients to request a service and then immediately execute other operations without waiting for the result ("fire and forget" model). Asynchronous capability is a crucial factor to make loosely coupled systems possible.

- Supports Remote Procedure Calls (RPCs)
  Web services enable clients to invoke methods and operations on remote objects using an XML-based protocol (SOAP). Input and output parameters which a web service must support are made available through remote procedures. A web service supports/implements RPC either by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

- *Supports document exchange*
  A central aspect of XML is that it is capable to represent data, simple and even complex documents in a generic way. This transparent exchange of documents *supported by Web services* contributes to make business integration easier.

2.5 Benefits of using Web Services

As also mentioned in [Pap08] and [Wha09], using web services offers many interesting benefits such as:

- Reusability
  A Web service is a component which can be remotely accessed using HTTP. Web Services provide a means to make a pre-existing code available through Internet. As a result, the program's functionalities can be invoked by other applications.

- Interoperability
  Web Services enable the share of data and the communication between different applications. For example, .NET applications can interact with Java web services and vice versa. Thus, applications become platform and technology independent.

- Standardized Protocol
  Web Services uses industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) use well defined protocol in the Web Services protocol stack. This standardization of protocol stack gives the business many advantages like wide range of choices, reduction in the cost due to competition and increase in the quality.

- Automatic Discovery
  Web Services automatic discovery mechanism allows businesses to easily find the

Scanned by CamScanner

Service Providers and retrieve web service description that have been previously published. The first step to access a web service is through the discovering process. Client queries the service registry for web service matching his needs. The query contains search criteria like service type or preferred price. After the discovery process is completed, the client can access the web service.

Scanned by CamScanner

what is cloud?

What is SaaS?

SaaS, or Software as a Service, describes any cloud service where consumers are able to access software applications over the internet. The applications are hosted in "the cloud" and can be used for a wide range of tasks for both individuals and organisations. Google, Twitter, Facebook and Flickr are all examples of SaaS, with users able to access the services via any internet enabled device. Enterprise users are able to use applications for a range of needs, including accounting and invoicing, tracking sales, planning, performance monitoring and communications (including webmail and instant messaging).

SaaS is often referred to as software-on-demand and utilising it is akin to renting software rather than buying it. With traditional software applications you would purchase the software upfront as a package and then install it onto your computer. The software's licence may also limit the number of users and/or devices where the software can be deployed. Software as a Service users, however, subscribe to the software rather than purchase it, usually on a monthly basis. Applications are purchased and used online with files saved in the cloud rather than on individual computers.

There are a number of reasons why SaaS is beneficial to organisations and personal users alike:

- **No additional hardware costs;** the processing power required to run the applications is supplied by the cloud provider.
- **No initial setup costs;** applications are ready to use once the user subscribes.
- **Pay for what you use;** if a piece of software is only needed for a limited period then it is only paid for over that period and subscriptions can usually be halted at any time.
- **Usage is scalable;** if a user decides they need more storage or additional services, for example, then they can access these on demand without needing to install new software or hardware.
- **Updates are automated;** whenever there is an update it is available online to existing customers, often free of charge. No new software will be required as it often is with other types of applications and the updates will usually be deployed automatically by the cloud provider.
- **Cross device compatibility;** SaaS applications can be accessed via any internet enabled device, which makes it ideal for those who use a number of different devices, such as internet enabled phones and tablets, and those who don't always use the same computer.
- **Accessible from any location;** rather than being restricted to installations on individual computers, an application can be accessed from anywhere with an internet enabled device.
- **Applications can be customised and whitelabelled;** with some software, customisation is available meaning it can be altered to suit the needs and branding of a particular customer.

Office software is the best example of businesses utilising SaaS. Tasks related to accounting, invoicing, sales and planning can all be performed through Software as a Service. Businesses may wish to use one piece of software that performs all of these tasks or several that each

Scanned by CamScanner

perform different tasks. The required software can be subscribed to via the internet and then accessed online via any computer in the office using a username and password. If needs change they can easily switch to software that better meets their requirements. Everyone who needs access to a particular piece of software can be set up as a user, whether it is one or two people or every employee in a corporation that employs hundreds.

Summary

- There are no setup costs with SaaS, as there often are with other applications
- SaaS is scalable with upgrades available on demand
- Access to Software as a Service is compatible across all internet enabled devices
- As long as there is an internet connection, applications are accessible from any location

Software-as-a-Service is software which is deployed over the internet and used by someone on a personal computer or local area network. Customers can often use software "pay-as-you-go" by using the cloud provider's licences, rather than buying licences themselves – which is why Software-as-a-Service is sometimes called "software on demand". As well as removing the need to buy, install and manage software at the user's end, it also has the advantage of the software being accessible from anywhere with an internet connection.

web services

Scanned by CamScanner

Messaging model used in WS

① **Synchronous Model**

Synchronous mean that every time a client access web service application, the client receives a SOAP response. Synchrous is req-response operation. Syn. services are designed when client appl^n require a more immediate response to a request. WS that rely on Synchronous communication are usually RPC-oriented. The java API for XML-based RPC (JAX-RPC) & SOAP with Attachments API for Java (SAAJ) uses the Synchronous model for Comm^n.

② **Asynchronous Model -**

Asynchronous means that the client which invokes a Web Service, does not or can not wait for a response. Thus asynch. is a one-way operation. The client send the req. in XML msg. The WS receives msg & process it, send result when it complete its process. An asyn. send requires a messaging provider which is not supported in SUN ONE Appl^n Server. Asy. receiver is supported in the appl^n server using the Java API for XML-msging (JAXM).

8, 9, 12, 18, 21, 24, 25, 26, 27, 30, 34, 37, 41, 43, 45, 47

Scanned by CamScanner

Assignment - No - 2

1. Explain WS - architecture with specific diagram.
2. Explain operations for the WS.
3. Explain the core building block of WS.
4. Define the following terms -
   1. Choreography 2. Service Orchestration.
   3. WS - context 4. Sej Service Reliability.
5. Explain basic steps of implementing Webservice.
6. Explain how to develop WS as enabled appln.
7. What are standard & technologies for implementing WS.

Assignment - 3.

1. Explain SOAP as a messaging protocol.
2. Explain SOAP msg structure.
3. Explain how to handle errors in SOAP based WS.
4. What are communication model for SOAP. Explain - RPC based & SOAP based commn model.
5. What are the advantages & disadvantages of 1 of the SOAP.
6. How to develope WS by using Java with using JAX-WS.

Scanned by CamScanner

1. What are the layers of web service protocol stack?
2. What is difference between REST Web services & SOAP Web Services?
3. Does Web Service call is synchronous or asynchronous?
4. What is XML RPC?
5. For Security purpose, what does a WS use in particular? - (Ans - SSL certificate)
6. What is REST?
7. What are the diff. browser that supports these web services?
8. What are the diff. channels / protocols are used for communication using WS?
   Ans - HTTP / GET, HTTP / POST / SOAP.
9. What are components of WS.
10. What's an example of real web service in action?
11. What types of operations are available in WSDL.
12. What are the steps to get a proxy object of a WS at the client side.

Scanned by CamScanner