**Web Services**

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)

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.

**Web Services - Architecture**

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.

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 XMLRPC 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.ise 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.