Web Service

Abstract:

The Internet has created the foundation for a networked economy—an extended business community in which vendors, partners, and customers interact and collaborate. But while the Internet Protocol (IP) standard has created a universal link between data and voice networks, companies use business applications built from a complex array of diverse platforms, operating systems, programming languages, and vendors. Now, the software industry wants to fix this with Web services. Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the web. Web services perform functions, which can be anything from simple requests to complicated business processes. We shall present the Motivation for Web services, Web services architecture: XML, SOAP, and WS-Addressing, Discovery, Current Challenges, Future Vision

Introduction:

Web services help disparate business applications, including those of trusted trading partners, work together more smoothly. The advantages are undeniable, and these software tools are quickly moving to the forefront of most technology executives' plans. Many in the business-technology industry trumpet Web services as a promising growth area. Web services allow companies to easily integrate their strategic applications with those of their partners, both internally and over the Internet. They can create cost-effective, flexible methods for conducting B2B transactions. In essence, Web services are network-based software applications that use extensible markup language (XML) to transmit, exchange, and interpret data among applications that would otherwise have difficulty interoperating. Companies of all sizes can employ Web services applications for a wide range of internal and external Internet-business functions, including supply-chain management and enterprise resource planning (ERP).

Web services have been promoted for several years, but how widespread are they today? What quantifiable benefits—and typical challenges—can a company expect when adopting Web services? When identifying the factors fueling interest in Web services, analysts point to the growing use of online B2B trading coupled with the increasing need to maximize existing network investments, boost productivity, and cut costs in a tough economy.

In a recent IDC report, nearly 50% of enterprises interviewed will have implemented Web services by 2006. IDC estimates that 5% of all U.S. enterprises had completed Web Services initiatives by the end of 2002, and it projects that 80% will have some kind of Web services project underway by 2008.

The Web has been a phenomenal success at enabling simple computer/human interactions at Internet scale. The original HTTP and HTML protocol stack used by today's Web browsers has proven to be a cost-effective way to project user interfaces onto a wide array
of devices. A key factor in the success of HTTP and HTML was their relative simplicity both HTTP and HTML are primarily text-based and can be implemented using a variety of operating systems and programming environments.

The World Web (WWW) is repository of information system spread all over the world linked together. The WWW has a unique combination of flexibility, portability and user-friendly feature that distinguish that it from other services provided by Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research. The WWW today is distributed client-server services, in which client using a browser can access services using a server. However, the provided is distributed over many locations called websites. In the early days of web, we had only non-interactive web sites, later interactive, and back end driven web sites. Businesses started using web sites to get a global presence. Back end driven web sites gave the concept of web based applications.

We can say Web services are another form of web applications, with self describing nature. When users request for a web application, it can serve them and can also connect with the other related web applications, and so, the users get all the services they need without knowing that they are served by different applications from different locations. Web services are nothing but applications with programmatic interface using certain standards and Internet - friendly protocols. Web services development, standards are the base and everything, to make them work.

Web services communicate using messages. They place a significant emphasis on how individual messages are formed and processed. Protocols are said to compose when they can be used either independently or in combination. Web services are autonomous agents whose development, deployment, operation, management, and security all vary independently from those of the service's consumer. The primary contribution of Web services has been the emphasis on protocol-based software integration. In Web services, a message is an XML document information item as defined by the XML Information Set. The set of possible information items generally maps to the various features in an XML document, such as elements, attributes, namespaces, and comments. Each information item has an associated set of properties that provide a more complete description of the item.

Security is a fundamental aspect of computer systems, especially those systems comprised of Web services. Security has to be robust and effective. Since systems may only make hard-wired assumptions about the format of messages and legal message exchanges, security has to be built based on explicit, agreed-upon mechanisms and assumptions. The security infrastructure should also be flexible enough to support the wide variety of security policies required by different organizations.
Benefits of Web Services:

Companies that develop and implement well-planned Web strategies enjoy far-reaching advantages. Leading companies strive to align their IT departments with their various business units, creating partnerships that can identify new opportunities and apply technology to achieve success. When the focus is on business agility, discretionary infrastructure and application development budgets are decentralized so that IT resources can be dedicated to those initiatives that bring the most competitive value to the company. At the same time, the primary objective for IT is to create a single, comprehensive, flexible Web platform that allows the business units to quickly implement and continually evolve their Web initiatives.

Network and Software Infrastructure

In just ten years, the Internet has evolved from an academic curiosity into an integral foundation for doing business. The deployment of web services for real-time business automation marks the culmination of that transformation into the rich fabric of today's Internet. Today, the Internet is more than just a physical network. The software layer of the World Wide Web is as much a part of the infrastructure as the underlying network components. Routers, fiber and cable provide a foundation of IP (Internet protocol) networking. Servers, databases and application platforms support the web-based software and services that run over the network Web services run over this software infrastructure. With their emergence, the Internet has ceased to be solely a content transmission network. It has become a Computing execution network, processing commercial transactions and business applications.

Universal Language

A strategically designed architecture will be built around the entire company's long-term goals, rather than the short-term goals of individual business units, and will be built with a view toward flexibility. This type of design will unify the enterprise's technology systems and facilitate the introduction of new applications and business initiatives. The ability to integrate applications hinges on the common adoption of extensible markup language (XML) and the creation of standard XML-based programming vocabularies used to translate between applications and documents. When all parties use these XML standards, applications can share data regardless of programming language, operating system, or device. The flexibility of XML and Web services gives users the option of choosing a particular hardware and software platform without being concerned about the integration cost. If priorities or market conditions change, a company can replace its Windows-based application with a Linux-based program and it will still work with the company’s Oracle database and individual user Windows applications. The software industry recognizes XML’s ability to bridge the gap between disparate systems, and
leading vendors are now marketing XML-ready platforms and developer tools that help customers create Web services, including Microsoft .NET framework, Sun Microsystems Open Net Environment (ONE), and IBM Web Services. Using software-development toolkits from these and other vendors, companies can customize their own Web services and use XML to translate between applications.

**Streamlined Integration**

Currently, Web services allow companies to use a common Internet standard to link previously incongruous internal systems or even connect with the systems of their partners and customers, thus increasing the overall value and functionality of the individual systems. For example, a company can create a Web service that allows information from its customer-relationship management (CRM) system to integrate into a sales or logistics program. Or it can use a Web service to allow its sales-forecasting program to interface with a supplier’s inventory-management system. Such flexibility comes from the ability to integrate with software components provided by trusted third parties, whether they are partners, customers, or vendors. Gartner expects Web services will dominate the deployment of new application solutions for Fortune 500 companies by 2004. E-integration technology underpins business processes that span two or more applications, both within and between enterprises. For example, when an e-integration system is in place, orders initiated on a company Web site can automatically go to the order management system without requiring a person to retype the information. Similarly, a product-availability query at the Web site will cause a message to be sent to the appropriate back-end systems in real time so the customer can get accurate, up-to-date information. Companies can extend this integration beyond their own enterprises to automate transactions with their trading partners.

The exclusive domain professionals are not needed. *Web content creation* has become a more user-friendly process, and more authoring tools have become accessible to the world at large. When authoring and publishing can be done by many people in the organization without a lot of specialized training, companies use content management systems to store information in a repository where it is easily accessible for reuse. This is of particular value in maintaining a strong, consistent brand across the e-business enterprise.

**Security Aspect**

As company units have become more integrated, and as customers have become users of the e-business enterprise's systems, *traditional security mechanisms have given way to "entitlement programs"*—modern security systems that intelligently manage and store user information and cross-reference it with enterprise access rules. These rules define entitlements in business terms, according to the role each person plays in each of the various business processes, providing an organization with the means to identify users and to securely provide them with the data and functionality they need, based on business requirements and policies.
Background:

Web services take many of the ideas and principles of the Web and apply them to computer/computer interactions. Like the World Wide Web, Web services communicate using a set of foundation protocols that share a common architecture and are meant to be realized in a variety of independently developed and deployed systems. Like the World Wide Web, Web services protocols owe much to the text-based heritage of the Internet and are designed to layer as cleanly as possible without undue dependencies within the protocol stack.

Technically the Web is Distributed Hypermedia systems that support interactive access. A hypermedia system provides a straightforward extension of traditional hypertext system. In either system information is stored as a set of documents. Besides the basic information a document can contain pointers to other document in the set.

The difference between a distributed and Non-distributed hypermedia system are significant. In non-distributed system, information resides in a single machine usually on single disk. In contrast web distributes documents across a large set of computer and a system administrator can choose to add remove, change or rename a document on a computer with out notifying other sites. Consequently links among web document are not always consistent.

A unit of hypertext or hypermedia available on the web is called Page. The main page of for an organization or an individual is known as homepage. Each web page that contains a hypermedia document uses a standard representation known as Hypertext Mark up Language, the standard allow an author to give general guide lines for display and to specify the contents of the page instead of specifying the detailed document format. A Web page is made up of two parts. Head is the first part of the web page, contains the title of the page and other parameters that the browser will use. The actual content of the web page are in Body, which include text and the tags. Here text is the actual information contained in a Page. The tags define the appearance of the document. The browser makes decision about the structure of the text based on the tags which are marks that are embedded into text. The beginning of the tag starts with the name of the tag and the ending tag start with slash followed by the name of the tag. The tag can have list of attributes, each of which can be followed by an equals sign a value associated with the attribute.

Conceptually, a Browser consists of a set of clients, a set of interpreters, and a controller that manages them. The controller forms the central piece of the browser. It interprets both mouse clicks and key board input and calls other component to perform operations specified by the user.

A Client that wants to access a Web page needs an address. To facilitate the access of documents distributed through out the world, HTTP uses the concept of Uniform Resource Locators. The URL is standard for specifying any kind of information on the internet. URL
defines four things, Method, host computer, and Path. Method is the protocol for the used to retrieve the document. Several different protocols can retrieve the document; among them are FTP and HTTP. The Host is the computer where information is located. The URL can optionally contain the Port number of the server. Path is the path name of the file where information is located. Although most current Version of HTTP is 1.1, HTTP version 1.0 and 0.9 are still in use.

Static Documents are fixed content documents that are created and stored in a server. The client can get only a copy of the document. In other words, the content of the file are determined when the file is created, not when it is used. Of course, the content in the server can be changed, but the user cannot change it. When the client access the document a copy of the document is sent. The user can then use a browsing program to display the document.

The Dynamic Document does not exist in predefined format. Instead a dynamic document is created by a web server, when ever browser requests a document. When a request arrives the web server runs an application program that creates the dynamic document. The server returns the output of the program as a response to the browser that requested the document.

For many applications we need a program to be run at the client site. These are called Active documents. For example, imagine we want to run a program that creates animated graphics on the screen or interacts with the user. The Program definitely needs to be run at the client site where the animation or interaction takes place. When a browser request an active document the server sends a copy of the document in the form of byte code. The document is then run at the client site. An active document in the server is stored in the form binary code. However it does not create overhead for the server in the same way that dynamic document does.

An important area in which Web services differ from the World Wide Web is scope. HTTP and HTML were designed around "read-mostly" interactive browsing of content that is often static, or at least highly cacheable. In contrast, the Web services architecture is designed for highly dynamic program-to-program interactions. In the Web services architecture, many kinds of distributed systems may be implemented. Examples include synchronous and asynchronous messaging systems, distributed computational clusters, mobile-networked systems, grid systems, and peer-to-peer environments. The broad spectrum of requirements in program-to-program interactions forces the Web services protocol stack to be much more general purpose than the first Web protocols.
Web services architecture

The core specifications used to formulate messages in the Web services architecture: XML, SOAP, and WS-Addressing.

The ideas behind Web services are starts with **XML**. XML -- the *eXtensible Markup Language* -- forms the backbone of the Web services to create and implement. XML is a markup language like HTML. XML has been designed to describe data and unlike HTML, XML tags are not predefined in XML. In Web services, a message is an XML document information item as defined by the XML Information Set. It is an abstract data model that is compatible with the text-based XML 1.0 and is the foundation of all modern XML specifications, XML Query, and XSLT 2.0. The set of possible information items generally maps to the various features in an XML document, such as elements, attributes, namespaces, and comments. Each information item has an associated set of properties that provide a more complete description of the item. There are eleven types of information items in an XML document.

In addition to the pure text-based encoding of the Infoset, the Web services architecture also supports an Infoset encoding that allows opaque binary data to be interleaved with traditional text-based markup. The W3C XML-binary Optimized Packaging (or XOP) format uses multi-part MIME to allow raw binary data to be included into an XML 1.0 document without resorting to base64 encoding. A companion specification, SOAP Message Transmission Optimization Method, or MTOM, [MTOM], then specifies how to bind this format to SOAP. XOP and MTOM are the preferred approach for mixing raw binary with text-based XML.

**SOAP**

SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML. SOAP was designed to reduce the engineering cost of integrating applications built on different platforms as much as possible with the assumption that the lowest-cost technology has the best chance of gaining universal acceptance. A SOAP message is an XML document information item that contains three elements: `<Envelope>`, `<Header>`, and `<Body>`.

The *Envelope* is the root element of the SOAP message and contains an optional Header element and a mandatory Body element. The *Header* element is a generic mechanism for adding features to a SOAP message in a decentralized manner. Each child element of Header is called a header block, and SOAP defines several well-known attributes that can be used to indicate who should deal with a header block (role) and whether processing it is optional or mandatory (must Understand). When present, the Header element is always the first child element of the Envelope. The *Body* element is always the last child element of the Envelope, and is a container for the "payload" intended for the ultimate recipient of the message. Tools that have been developed for the XML data model may be used for inspecting and constructing complete messages.
Any software agent that sends or receives messages is called a *SOAP node*. The node that performs the initial transmission of a message is called the *original sender*. The final node that consumes and processes the message is called the *ultimate receiver*. Any node that processes the message between the original sender and ultimate receiver is called an *intermediary*. Intermediaries are used to model the distributed processing of an individual message. The collection of intermediary nodes traversed by the message and the ultimate receiver are collectively referred to as the *message path*.

To allow parts of the message path to be identified, each node participates in one or more roles. The base SOAP specification defines two built-in roles: *Next* and *UltimateReceiver*. Next is a universal role in that every SOAP node other than the sender belongs to the Next role. UltimateReceiver is the role that the terminal node in a message path plays.

*The body of a SOAP envelope is always targeted at the ultimate receiver.* In contrast, *SOAP headers may be targeted at intermediaries or the ultimate receiver.* To provide a safe and versionable model for processing messages, SOAP defines three attributes that control how intermediaries and the ultimate receiver process a given header block—role, relay, and mustUnderstand. The role attribute is used to identify which node the header block is targeted at. The mustUnderstand attribute indicates whether that node may ignore the header block if it is not recognized. Header blocks marked mustUnderstand="true" are called mandatory header blocks. Header blocks marked mustUnderstand="false" or that have no mustUnderstand attribute are called optional header blocks. The relay attribute indicates whether that node should forward unrecognized optional headers or discard them.

**Message Exchange Patterns**

The messaging flexibility provided by SOAP allows services to communicate using a variety of message exchange patterns, satisfying the requirements of distributed applications.

The use of remote procedure calls, for example, popularized the synchronous request/response message exchange pattern. When message delivery latencies are uncontrolled, asynchronous messaging is needed. When the asynchronous request/response pattern is used, explicit message correlation becomes mandatory.

Broadcast transports popularized one-to-many message transmissions. The original sender imposing its messages on the recipients by just sending them is referred to as the push model. While this model is effective in local-area networks, it does not scale well to wide-area networks nor offer recipients an option to regulate the message flow.

Another useful pattern is based on an application's ability to express interest in particular kinds of messages, making the publish/subscribe pattern quite popular. By explicitly
subscribing to message sources (or topics), applications have a somewhat more controlled flow relevant information.

The pull model is used when a recipient explicitly requests a message from a source. This makes message flow the recipient's responsibility. The pull pattern can also be combined with publish/subscribe. It is well suited for situations where recipients may be intermittently disconnected from the sources.

**Transport Independence**

Since Web service protocols are designed to be completely independent of the underlying transport, selection of the appropriate mechanism can be deferred until runtime. This allows Web service applications the flexibility to determine the appropriate transport as the message is sent. Additionally, the underlying transport may change as the message is routed between nodes, and again, the mechanism selected for each hop can vary as required.

Despite this general transport independence, most first-generation Web services communicate using HTTP, as this is one of the primary bindings included within the SOAP specification. HTTP uses TCP as its underlying transport protocol. However, TCP's design introduces processing overhead this is not always necessary.

Several application protocol patterns more closely match the semantics of the User Datagram Protocol, or UDP. These patterns are particularly useful for devices and other resource-constrained systems. UDP does not have the delivery guarantees of TCP; it provides best-effort datagram messaging. It also requires fewer resources to implement than TCP. In addition, UDP provides multi-cast capabilities, allowing a sender to simultaneously transmit a message to multiple recipients. The specifications for binding SOAP messages to UDP are published in *SOAP-over-UDP*.

**WS-Addressing**

The WS-Addressing specification defines three sets of SOAP header blocks for this purpose.

The *Action header* block is used to indicate the expected processing of a message. This header block contains a single URI that is typically used by the ultimate recipient to dispatch the message for processing.

The *MessageID* and *RelatesTo* header blocks are used to identify and correlate messages. The MessageID and RelatesTo headers use simple URIs to uniquely identify messages—typically these URIs are transient UUIDs.
The To / ReplyTo / FaultTo header blocks are used to identify the agents that are to process the message and its replies. These headers rely on a WS-Addressing-defined structure called an endpoint reference that bundles together the information needed to properly address a SOAP message.

Metadata

To provide for a robust development and operational environment, services are described using machine-readable metadata. Metadata enables interoperability. Web service metadata serves several purposes. It is used to describe the message interchange formats the service can support, and the valid message exchange patterns of a service. Metadata is also used to describe the capabilities and requirements of a service.

The Web Service Description Language

The Web Service Description Language, or WSDL, was the first widely adopted mechanism for describing the basic characteristics of a Web service. A WSDL description is a first step in automatically identifying all characteristics of the target service and enabling software development tools. In addition to describing message contents, WSDL may define where the service is available and what communications protocol is used to talk to the service. This means that the WSDL file can specify the base elements required to write a program to interact with a Web service. Several tools are available to read a WSDL file and generate the code required to produce syntactically correct messages for a Web service.

WS-Policy

WS-Policy provides a general-purpose model and syntax to describe and communicate the policies of a Web service. It specifies a base set of constructs that can be used and extended by other Web service specifications to describe a broad range of service requirements and capabilities. WS-Policy introduces a simple and extensible grammar for expressing policy assertions and a processing model to interpret them. Assertions may be combined into logical alternatives.

Discovery

Web service discovery is a key enabler for automating connections to services without human intervention. The Web service approach to discovery mirrors the two most common approaches to finding information in a computer system: looking in a well-known location, or broadcasting a request to all available listeners. The UDDI registries serve as the directory, and discovery protocols are used to broadcast requests.

UDDI directory

The Universal Description, Discovery, and Integration protocol, or UDDI, specifies a protocol for querying and updating a common directory of Web service information. The directory includes information about service providers, the services they host, and
the protocols those services implement. The directory also provides mechanisms to add metadata to any registered information.

The UDDI directory approach can be used when Web service information is stored in well-known locations. Once the directory is located, a series of query requests can be sent to obtain the desired information. UDDI directory locations are obtained out of band, usually through system configuration data.

Web service providers have various options for how they deploy UDDI registries. Deployment scenarios fall into one of three categories: public, extra-enterprise and intra-enterprise. To support public deployments, a group of vendors led by Microsoft, IBM and SAP host the UDDI Business Registry. The UBR is a public UDDI registry that is replicated across multiple hosting organizations, serving as a resource for Internet-based Web services.

For all deployment scenarios, UDDI directories contain detailed information about Web services and where they are hosted. A UDDI directory entry has three primary parts – the service provider, Web services offered, and bindings to the implementations. Each of these parts provides progressively more detailed information about the Web service.

The most general information describes the service provider. This information is not targeted at Web services software, but at a developer or implementer that needs to contact someone responsible for the service directly. Service provider information includes names, addresses, contacts and other administrative details. All UDDI entries have multiple elements for multi-language descriptions.

The list of available Web services is stored within a service provider entry. These services may be organized depending on their intended use: they may be grouped into application area, geography, or any other scheme that is appropriate. Service information stored in a UDDI registry includes simply a description of the service and a pointer to the Web service implementations it contains. Links to services hosted by other providers, called 'Service Projections', may also be registered.

The final part of a UDDI service provider entry is the binding to an implementation. This binding associates the Web service entry to the exact URI(s) to identify where the service is deployed, specifies the protocol to use for access, and contains references to the exact protocols that are implemented.

Dynamic Discovery

Dynamic Web service discovery is provided in a different manner. As an alternative to storing information in a known registry, dynamically discovered Web services explicitly announce their arrival and departure from the network. WS-Discovery defines protocols to announce and discover Web services through multicast messages.
When a Web service connects to a network, it announces its arrival by sending a Hello message. In the simplest case, these announcements are sent across the network using multicast protocols – we call this an ad-hoc network. This approach also minimizes the need for polling on the network. In order to limit the amount of network traffic and optimize the discovery process, a system may include a Discovery Proxy. A Discovery Proxy replaces the need to send multicast messages with a well-known service location, transforming an ad-hoc network into a managed network. Using configuration information, collections of proxy services may be linked together to scale the discovery service to groups of servers, scaling from one machine to many.

Since the Discovery Proxies are themselves Web services, they may announce their presence with their own special Hello message. Web services receiving this message may then take advantage of the proxy's services, and are no longer required to use the noisier one-to-many discovery protocols.

When a service departs from a network, WS-Discovery specifies a Bye message to be sent to either the network or the Discovery Proxy. This message informs the other services on the network that the departing Web service is no longer available.

**Agreement Coordination Protocols – Reliable Messaging and Transactions**

When multiple Web services must complete a joint unit of work or operate under a common behavior, there must be common agreement on what protocols to use. This minimum amount of coordination among Web services is unavoidable. Coordination protocols are also required to be able to determine and to agree that a common goal has been reached. Every interaction between Web services can be viewed as a kind of coordination. Agreement coordination protocols bring the architecture an improved chance that the participant services will succeed in what they set up to do jointly. The Web services architecture is designed to function properly in the face of transports that lose messages and services that malfunction.

The specifications that define this functionality are *WS-ReliableMessaging, Reliable Messaging WS-Coordination, WS-AtomicTransaction and WS-BusinessActivity.*

Many conditions may interrupt an exchange of messages between two services. This is especially an issue when unreliable transport protocols such as HTTP 1.0 and SMTP [SMTP] are used for transmission or when a message exchange spans multiple transport-layer connections. Messages may be lost, duplicated or reordered, and Web services may fail and lose volatile state. *WS-ReliableMessaging* is a protocol that
enables the reliable delivery of messages based on specific delivery assurance characteristics. The specification defines three different assurances that may be used in combination:

At-Least-Once Delivery: Each message is delivered at least one time.
At-Most-Once Delivery: Duplicate messages will not be delivered.
In-Order Delivery: Messages are delivered in the same order they were sent.

The combination of at-least-once and at-most-once assurances results in an exactly-once delivery assurance. Due to the transport-independent design of the Web services architecture, all delivery assurances are guaranteed irrespective of the communication transport or combination of transports used. Using WS-ReliableMessaging simplifies system development due to the smaller number of potential delivery failure modes that a developer must anticipate.

Designated Coordinators
Some families of N-way coordination protocols require a designated coordinator to shepherd a unit of work through a number of cooperating services. One example is when activities must be coordinated between services that are not all expected to be connected at the same time. As long as each participant and the coordinator communicate at some time, coordination may happen and agreement on the outcome may be reached. The Web services architecture defines some simple operations for designated coordinators

Within the context of the atomic transaction coordination type, three protocols are defined: a Completion protocol, and two variants of a Two-Phase Commit protocol. The Completion protocol is used to initiate commit processing. A Web service registered for Completion has the ability to tell the designated coordinator when commit processing is to begin. This protocol also defines messages to communicate the final result of the transaction to the initiator. However, the protocol does not require that the coordinator ensure that the result was processed by the initiator. In contrast, other behaviors in WS-AtomicTransaction do require the coordinator to ensure that participants process the coordination messages.

WS-BusinessActivity specifies two protocols for long-running transactions. Instead of holding locks on resources until the transaction is committed, the WS-BusinessActivity specification is based on compensating actions.

Enumeration, Transfer, and Eventing
These specifications provide enumeration of service resources, their state management, and event notification in the Web services architecture. They are based on WS-Enumeration, WS-Transfer, and WS-Eventing
Enumeration

Many scenarios require data exchange using more than just a single request/response message pair. Types of applications that require these longer data exchanges include database queries, data streaming, the traversal of information such as namespaces, and enumerating lists. Enumeration, in particular, is achieved through establishing a session between the data source and the requestor. Successive messages within the session transport the collection of elements being retrieved. No assumptions are made on the approach used by the service to organize the items that will be produced. What is expected is that under normal processing circumstances the enumeration will produce all the underlying data before the end of the session. WS-Enumeration specifies protocols to establish an enumeration session and to retrieve sequences of data. WS-Transfer introduces operations that create, update, retrieve and delete resources. WS-Eventing specifies mechanisms that allow a Web service, also referred to as a subscriber, to register interest in specific events that are provided by another Web service (the event source). This registration is called a subscription. WS-Eventing defines operations an event source can provide that allow subscriptions to be created and managed.

Current Challenges

Many questions must be answered before the Web-services picture painted by technology vendors becomes a reality. Who decides the standards for the vocabularies? How will vendors provide end-to-end security? Who is accountable when vendors fail to deliver promised services? If customers are relying on hosted Web services for critical business functions, how will vendors guarantee availability and reliability? How will vendors handle pricing? These are formidable issues, but technology vendors are determined to address them and are displaying a rare level of cross-industry cooperation to drive Web-services adoption.

Security Challenges

Security professionals tend to focus narrowly on protecting assets, often at the expense of business agility. Frustrated business users bypass security procedures and mechanisms to complete tasks. As company units have become more integrated, and as customers have become users of the e-business enterprise's systems, traditional security mechanisms have given way to "entitlement programs"—modern security systems that intelligently manage and store user information and cross-reference it with enterprise access rules. These rules define entitlements in business terms, according to the role each person plays in each of the various business processes, providing an organization with the means to identify users and to securely provide them with the data and functionality they need, based on business requirements and policies. Presently Security professionals help to rapidly launch new business activities while protecting assets. They now focus on making it as quick and easy
as possible for the company to establish new relationships and for authorized people to complete tasks without compromising key assets. To maximize the reach of Web services, end-to-end security must be provided when intermediaries are not trusted by the communicating endpoints. This requires higher-level security protocols. End-to-end message security is a richer alternative to point-to-point transport-level security, since it supports the loosely coupled, federated, multi transport, and extensible environment that SOAP-based Web services require. This powerful and flexible infrastructure can be developed from a combination of existing technologies and Web services protocols while mitigating many of the security risks associated with point-to-point messaging. Even though the security requirements for Web services are complex, no new security mechanisms were invented to satisfy the needs of SOAP-based messaging. Web services security is based on the requirement that incoming messages prove a set of assertions made about a sender, a service or other resource. We call these claims, or security assertions. Examples of security claims include identity, attributes, key possession, permissions, or capabilities.

**High Availability Web Services Using Cluster**

Administrators of large Web sites are faced with managing the growth in a 24x7 environment. This requires scalable solutions that do not require system downtime, yet effectively manage the workload across multiple servers. Enterprises are implementing Internet business strategies to increase revenue, raise customer satisfaction levels, streamline their supply chain management, and optimize their workforce. Consequently, enterprises are experiencing ever-increasing demands from customers, suppliers, and employees for access to applications and data. These challenges are being met by a number of networking strategies and services, and server clustering frequently provides an optimum solution. Server clustering is accomplished by giving multiple real servers the appearance of a single virtual server. IP traffic destined for the virtual server is assigned to the server in the cluster that is best suited for that transaction.

A server cluster is a set of computer systems connected together through multi system hardware or software to provide services that were traditionally provided by a single system. Clustering of servers provides the following benefits:

*Rapid response to unexpected growth*

Servers can be added to the cluster without disrupting the work that is executing on the other servers. When a server is put into service, the work will start to migrate to that server, thus reducing the load on the existing servers.

*Balanced multiple work-load*

Spreading users across multiple independent systems can result in wasted capacity on some systems while others are overloaded. By employing intelligent load balancing within a cluster of systems, the users are spread to available systems based on the load of each system.
Continuous application availability
Individual application instances can be taken down or fail without shutting down service to end users. Users of the application taken down can immediately reconnect to an alternate system, unaware that they are connected to an alternate server. Users on the other servers are virtually unaffected except for the additional load caused by services for some portion of the users that were formerly on the failed system.

Investment protection
Existing servers and applications are not replaced by alternatives as requirements grow. Additional “footprints” can be added to the cluster, which saves hardware and software costs and does not impact

Server clustering is accomplished by giving multiple real servers the appearance of a single virtual server. IP traffic destined for the virtual server is assigned to the server in the cluster that is best suited for that transaction. The best-suited server would be one that is active and in service, the IP application is running, and the server is the least busy of the servers in the server cluster. As a result, client requests are effectively distributed across the server cluster to the server that is most capable of responding to the request.

Future Vision
Web services essentially make the Internet itself the basis of a new Operating System, and this has software vendors excited about the future. Most businesses are now connected to the Internet, and the long-term vision for Web services goes beyond simply integrating existing programs to delivering plug-and-play software applications on demand over the Internet. Technology vendors plan to develop, market, and lend online Web services to fulfill virtually any business function. Companies will be able to simply search a public directory of applications and download those that fit their needs. This process would involve minimal integration or deployment headaches because all Web services will use the same information-sharing standard. Technology vendors are saying that users will have fewer problems upgrading or troubleshooting applications. Because the applications are sold as Web-based services, providers can minimize piracy and customize and differentiate software depending on customer needs. Web services promise all this, along with reduced deployment and integration costs. A few examples now are like Stock price information gathering, whether reports, Flight reservations etc., As the technology explores many things are possible through web. Right now we have only tools and standards which are still not matured. It is yet to be simplified, and till then we have to work only with the technologies which are yet to improve a lot. But its needless to say that its time to learn and practice some web services development.