WS-BPEL

BPEL (Business Process Execution Language) is an XML-based language that allows web services, APIs and human processes in a service-oriented architecture (SOA) to interconnect and share data in a business workflow.

BPEL is based on XML and defines the model and the means to describe the behavior of the process and allows the execution of processes, which consist of calls of web services, which is referred to as the **service orchestration**. BPEL language was developed by Microsoft and IBM, and was standardized by a consortium OASIS (Organization for the Advancement of Structured Information Standards).

BPEL Language in practice: BPEL language is perfectly suited to a clear description of the processes by which processes can be then **mechanically exercised**. BPEL enables organizations to automate their processes (so-called service orchestration). BPEL language allows description of the conduct and behavior of business processes as a sequence of activities, including their branches, which are engaged in the process, while these activities are represented by web services (WS).

The basic features include principles of **structured programming** (conditions, cycles, etc.), the possibility of error handling, support for long lasting processes, etc. Lack of BPEL is the absence of standard graphical representation. For this purpose, the BPMN notation is often used, but the mapping between the two languages is not clear. BPEL has become in recent years an important standard that helps to use SOA, not only at the level of informatics, i.e. by IT architects, but also at the level of organizational processes. Process in the WS-BPEL standard describes the sequence and conditions of service call in a service-oriented architecture (SOA), also known as the orchestration of services.

BPEL choreography vs. orchestration

BPEL offers two paradigms of operation: choreography or orchestration.

Orchestration is now the primary way to implement BPEL.

In choreography there is no central program that manages BPEL processes. Each service directly communicates with the other services it relies on without central authority. This can be thought of like a choreographed dance, each dancer knows their role and how they fit in their place, so they don't collide, but there is no director on the stage during the performance.

While choreographed systems can be simple to set up in small deployments, they can quickly grow unwieldy when many pieces are added and are more difficult to integrate new processes or to change the underlying services. Therefore, orchestrated BPEL implementations are now the standard.

In an orchestrated system, a central program manages the BPEL processes and communicates with the services. Each service does not need to be aware of the other services, only the central director. This can be thought of like an orchestra playing music.

Each instrument only has its own sheet music to read from. The conductor is on stage directing each section and ensuring that they work together as a whole. Orchestrated systems often require additional setup but are more agile and resilient.

WS_Security

Security concerns are threatening to dampen the widespread usage and adoption of Web services-related technology—in the enterprise scale. Conflicting sets of standards, technologies, and specifications have emerged—with no clear-cut direction or leadership.

Enterprise Security

Let's try to categorize all the generic security concerns in any enterprise-computing realm. Later, we will be able to discuss their relevance and importance in service-driven enterprise architectures.

Securing Access to Resources

Every application that services a client application—be it an enterprise data warehouse or a back office application—can be called as a resource. When a client makes a request to the resource to perform a specific operation, the resource needs to ensure that

- The client has valid access to the resource (Authentication)
- The client has valid privileges to perform the operation (Authorization)

Methodologies to Implement Authentication

- Username and password: The most fundamental form of authentication that is widely adopted across all applications, including the Web.

- Concept of secure ID: Usernames and passwords have the threat of being shared across several individuals. Hence, for restricted business applications, a secret PIN number is often associated with the username during authentication. This PIN number usually gets generated in a physically secure ID card, and keeps changing every minute or so. The user has to refer to the active PIN number in the card at any given point of time.

Methodologies to Implement Authorization

- Access control lists: After the authentication is successful, we need to find out whether the requesting application has the necessary privileges to perform the requested operation. This is usually accomplished by referring to a table, called the access control list, which lists all the privileges against the given username.

Securing the Message (Data)

Confidential business data gets exchanged across the network—within and outside the enterprise system boundaries. It is important to ensure that:

- Any unauthorized process that has access to the packets that travel across the network is not able to decipher the data exchanged (Confidentiality)

- No modifications are made to the original message or data while it is traveling across the network (Integrity)

- The sender application should not be able deny its role in sending the data at a later date (during disputes, for example). (Non-repudiation)

Methodologies to Implement Confidentiality

- Encryption and cryptography: Cryptographic algorithms have played a pivotal role in driving secure messaging to new heights, and continue to inspire technologies, even today. The methodology involves converting data to meaningless chunk characters (encrypted data) using complex numeric transformations and secret keys, and deciphering the same at the receiver end using reverse algorithms.

Many cryptographic technologies and toolkits are available in the markets today. They help applications to encrypt/decrypt the data on either end, without bothering much about the nuisances of algorithms.

Methodologies to Implement Integrity and Non-repudiation

- Digital signatures: Digital signatures are human signature's counterpart in the electronic world. Though they are built on the top of cryptography, they serve a slightly different purpose—attesting the original source of the message (or data) as well as ensuring that the data has not been tampered with while in transit in the network.

The basic concept behind digital signatures is the ownership of secret keys: public keys (which are circulated to everyone) and private keys (which are held by specific systems).

Securing the Wire (Network)

Securing the network, which establishes the physical communication layer across the systems, involves setting up routers, firewalls, and sub-domains.

Impact of Web Service Interactions in the Security Realm

Web service technologies provide the much-required flexibility in application-to-application integration. This provides room for complex interactions across disparate systems scattered along the network and the Web.

- Web services promote many-to-many models (several services interacting with several others) instead of the many-to-one model (several clients interacting with one server). This has serious impact on the security methodologies that have been adopted so well in the past. For example, in a client-server model, it is enough if we authenticate and authorize a person or application during login process; and as long as the session is valid, the server-side applications can continue to serve the requests. But the same is not applicable in case of Web services because the services themselves may not be available in a single server or network. Sharing session and authentication information across the network—across disparate application—is not only difficult, but resource-intensive, as well. Suppose that one client request associated with five different services—authenticating and authorizing each time—results in 10 database calls. If the client makes five requests, it results in 50 database calls! Even if we were to appoint a dedicated service for authentication/authorization in front of the business service, each service would have to validate the session information with this security service, which is highly resource-intensive.

- The level of trust placed over a given client request might vary across different services. For example, a request from the company's employee through an intranet might enjoy more privileges than a similar request from a business partner through the Internet. When each of these Web services tend to evolve their own privilege matrix, it becomes difficult to assemble them to evolve more meaningful services. This complexity has inspired companies to evolve a separate trust layer like the Microsoft's Trust Framework.

- As service federations (pools of related services) evolve with the promotion of service-driven architectures, it becomes important to distinguish a peer service's request from that of an outsider. This means the evolution of common security infrastructure for all federal services in the domain and shared privileges and access rights (Federated identity). The grammar security architecture will depend upon the complexity of interactions within the federation. An even more complex scenario is "dynamic Web service federations," in which service federations emerge just in time, based on the nature of client requests!

- As Web services result in highly decentralized application interactions, it becomes difficult to share and distribute secret cryptographic keys. As a direct result, messaging security layers that ensure integrity, confidentiality, and non-repudiation of a given request are all affected. Ensuring a safe pipe from end to end becomes complex.

- Web services technologies are touted as platform-independent and language neutral standards, relying solely on XML and its offshoots at their base. This means we cannot readily make use of the existing/proven security solutions in the market that are available in different flavors for different runtime environments to secure Web services. Many of them, being platform-specific, may not be flexible enough to extend themselves to an XML-driven security architecture.

Web Services Security: Technologies and Standards

Industry-neutral bodies and consortia—such as W3C (http://www.w3c.org), IETF (http://www.ietf.org), and OASIS (http://www.oasis.org)—propel Web services technologies and standards. Individual business entities such as Microsoft and IBM submit their technologies as proposals to these bodies, where they undergo suitable revisions by participating members before being recommended as an industry standard. Core Web services technologies such as SOAP and WSDL evolved in this way.

Although such efforts have promoted platform- and language-independent standards, they have also slowed down the overall progress and growth of Web services technologies. Evolving consensus across competing standards and business bodies becomes difficult because each company wants to take competitive advantage over the others by making its own technology a W3C standard. Some companies have even forged independent partnership deals and alliances with their close allies in dictating Web service security standards, leaving the rest behind.

All these issues have slowed down the evolution of Web service security standards and technologies. A genuine enterprise developer is left with two choices for securing his Web service: Wait for the standards to evolve or go ahead with proprietary mechanisms dominated by vendor-specific security frameworks.

Despite this gloomy scenario, a few technologies have established themselves in the Web services security realm, and are well worth being considered as industry standards, including the following:

- Security Assertion Markup Language (SAML) for Authentication/Authorization

- XML Access control markup language (XACML) for Authentication/Authorization

- XML Key Management Specifications (XKMS) for Cryptography

- XML Encryption for Confidentiality

- XML Digital Signatures for Integrity and Non-repudiation

- SOAP security extensions for Integrity and Non-repudiation