

## Description

A Web Service is programmable application logic accessible using standard Internet protocols. Web Services combine the best aspects of component-based development and the Web. Like components, Web Services represent black-box functionality that can be reused without worrying about how the service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-specific protocols, such as DCOM, RMI, or IIOP. Instead, Web Services are accessed via ubiquitous Web protocols (ex: HTTP) and data formats (ex: XML).

The software industry is finally coming to terms with the fact that integrating software applications across multiple operating systems, programming languages, and hardware platforms is not something that can be solved by any one particular proprietary environment. Traditionally, the problem has been one of tight-coupling, where one application that calls a remote network is tied strongly to it by the function call it makes and the parameters it requests. In most systems before Web services, this is a fixed interface with little flexibility or adaptability to changing environments or needs.

Web services uses XML that can describe any and all data in a truly platform-independent manner for exchange across systems, thus moving towards loosely-coupled applications. Furthermore, Web services can function on a more abstract level that can reevaluate, modify or handle data types dynamically on demand. So, on a technical level, Web services can handle data much easier and allow software to communicate more freely.

On a higher conceptual level, we can look at Web services as units of work, each handling a specific functional task. One step above this, the tasks can be combined into business-oriented tasks to handle particular business operational tasks, and this in turn allows non-technical people to think of applications that can handle business issues together in a workflow of Web services applications. Thus, once the Web services are designed and built by technical people, business process architects can aggregate them into solving business level problems. To borrow a car engine analogy, a business process architect can think of putting together a whole car engine with the car frame, body, transmission, and other systems, rather than look at the many pieces within each engine. Furthermore, the dynamic platform means that the engine can work together with the transmission or parts from other car manufacturers.

What rises from this last aspect is that Web services are helping to bridge the gap between business people and technologists in an organization. Web services make it easier for business people to understand technical operations. Business people can describe events and activities and technologists can associate them with appropriate services.

With universally defined interfaces and well designed tasks, it also becomes easier to reuse these tasks and thus, the applications they represent. Reusability of application software means a better return on investment on software because it can produce more from the same resources. It allows business people to consider using an existing application in a new way or offering it to a partner in a new way, thus potentially increasing the business transactions between partners.

Therefore, the primary issues that Web services tries to tackle are the issues of data and application integration, and that of transforming technical functions into business-oriented computing tasks. These two facets allow businesses to communicate on a process or application level with their partners, while leaving dynamic room to adapt to new situations or work with different partners on demand.