1) What is Web Service?

Web Service is a software system for communicating two devices over the network. [More details...](http://www.javatpoint.com/what-is-web-service)

2) What are the advantages of web services?

* **Interoperability**: By the help of web services, an application can communicate with other application developed in any language.
* **Reuability**: We can expose the web service so that other applications can use it.
* **Modularity**: By the help of web service, we can create a service for a specific task such as tax calculation etc.

[More details...](http://www.javatpoint.com/what-is-web-service)

3) What are the different types of web services?

There are two types of web services:

* SOAP
* RESTful

4) What is SOAP?

SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services. [More details...](http://www.javatpoint.com/soap-web-services)

5) What are the advantages of SOAP web services?

* WS Security
* Language Independent
* Platform Independent

[More details...](http://www.javatpoint.com/soap-web-services)

6) What are the disadvantages of SOAP web services?

* Slow
* WSDL Dependent

[More details...](http://www.javatpoint.com/soap-web-services)

7) What is WSDL?

WSDL stands for Web Services Description Language. It is a xml document containing information about web services such as method name, method parameter etc. [More details...](http://www.javatpoint.com/web-service-components)

8) What is UDDI?

UDDI stands for Universal Description, Discovery and Integration. It is a XML based framework for describing, discovering and integrating web services. It contains a list of available web services. WSDL is the part of UDDI. [More details...](http://www.javatpoint.com/web-service-components)

9) What is RESTful web services?

REST stands for REpresentational State Transfer. It is a architectural style. It is not a protocol like SOAP. [More details...](http://www.javatpoint.com/restful-web-services)

10) What are the advantages of RESTful web services?

* Fast
* Language Independent
* Platform Independent
* Can use SOAP.
* Allows different data format.

[More details...](http://www.javatpoint.com/restful-web-services)

11) What is the difference between SOAP and REST web services?

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| 6) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |

[More details...](http://www.javatpoint.com/soap-vs-rest-web-services)

12) What is SOA?

SOA stands for Service Oriented Architecture. It is a design pattern to provide services to other application through protocol.[More details...](http://www.javatpoint.com/service-oriented-architecture)

13) What tools are used to test web services?

* **SoapUI tool** for testing SOAP and RESTful web services
* **Poster** for firefox browser
* **Postman** extension for Chrome

|  |  |  |
| --- | --- | --- |
| Item | **SOAP** | **REST** |
| 1 | Most vendors are endorsing or supporting SOAP meaning that security issues will be readily exposed and are likely to be fixed or addressed through countermeasures, patches, etc.  Some SD tool vendors and servers are supporting both REST and SOAP (among other options). | REST is widely endorsed by developers and has a strong historical foundation; although REST specific security guidance has been limited (there is no “leader” for this, such as the W3C with SOAP). |  |
| 2 | SOAP messages can be (partially) secured by following/implementing WS-Security (WSS). This allows for integrity and confidentiality controls.  The W3C is also issuing security guidance with respect to SOAP and there is some expectation that WSS will be enhanced both by the W3C and by vendors. | REST, dependent on HTTP, does not support secure messaging directly. HTTPS (SSL or TLS with HTTP) is normally proposed, requiring another port to be opened in the Firewall. Also, this approach can make it difficult to proxy sessions. A number of HTTPS related vulnerabilities are known, across multiple products.  HTTPS is only a point to point solution and (mainly in theory) it can be compromised. Use of longer (e.g. 128 bit or larger) keys limits the likelihood of data interception/decryption.  Both SOAP and REST require many additional security supplements to address all issues (e.g. authentication), beyond simply the protection of message content and integrity.  Perhaps most importantly, most security supplements (including HTTPS) only protect communications and attacks on the server are still a problem. |  |
| 3 | SOAP formatted XML envelopes and XML in general are prone to manipulation (e.g. “infinite string attacks”). Many XML vulnerabilities and exploits are known. |  |  |
| 4 | SOAP, when implementing HTTP transits network firewalls, through port 80 (or 8080) effectively tunnelling through that level of security. This has been touted as a big SOAP benefit for developers who can avoid asking for extra ports to be opened on client/server side firewalls (i.e. for other protocols than HTTP).  However, this is a problem from an infrastructure perspective. The most consistent issue is the difficulty of deciding what to let through and what to block.  Having multiple protocols running over one port makes filtering much harder.  Intrusion Detection/Prevention Solutions (IDS/IPS) are used to examine incoming this traffic in close detail.  But the more complex the traffic, the longer the IDS examination takes (in milliseconds). And, SOAP seems to be quite complex (e.g. multiple headers) and message structure can be contextual.  Decisions would need to be made (in advance) as to a “standard” SOAP header structure, protocols (HTTP is likely to be the only one which systems staff will allow vs. SMTP, for example) etc. to be able to block/pass SOAP messages at the Firewall and at the IDS.  More importantly, when SOAP uses HTTP, there is no built-in way for the Firewall (or an IDS) to know if messages are coming from an authenticated source. | REST is tied to HTTP or HTTPS which means that there is more visibility (e.g. HTTP 1.1) to messages which is an advantage for systems staff who have to configure Firewalls and/or IDS solutions to inspect traffic coming into or leaving the secure perimeter.  This would make it easier to route between multiple Web services.  All of the above are also well understood by IT systems staff, with respect to:   * Protocol format * Port usage * Traffic patterns * Contextual issues (e.g. when a request seems “normal”) which can be significant if incident intervention is (potentially) needed |  |
| 5 | An attacker can send a specially crafted SOAP request, which makes use of parameter entities to inflict a denial of service condition on the server. This can drive the server to 100% loading resulting in a DOS condition.    This is a known issue with various SOAP compatible servers (patches exist). |  |  |
| 6 |  | Access Control Lists (ACL) can readily be applied to REST traffic (e.g. to restrict POST requests to certain users or IP source addresses). In practice this is considered ineffective. |  |
| 7 |  | Local HTTP caching presents security challenges which should be addressable (e.g. through cache clearing) but which will need to be tested. |  |
| 8 |  | Logging (for audit trails) is simple with REST as all relevant information is held in the URI. |  |
| 9 | Potential SOAP exploits are less well understood as usage is still somewhat limited (although growing fast). Solutions are potentially complex. However, the WS-Security standard is an effort to provide a fairly open (hence upgradeable) standard to augment SOAP (e.g. for authentication of senders, protection of message integrity, etc.). | REST operations (e.g. GET, PUT) and associated exploits are quite well understood and documented (e.g. flooding of HTTP GET requests, HTTP GET request with long parameter names triggering overflows, etc.) – as are solutions. |  |
| 10 |  | Per item 9, if  GETs are accepted from anyone, some or all POSTs can be blocked. If some POSTs are to be allowed (e.g. by a group of trusted users), Access Control Lists (ACL) can be applied.  There are various ways to do this. However, ACLs require manual maintenance (e.g. IP address changes or user profile changes).  This would also require the creation (and enforcement) of strong developer guidelines to prevent un-managed use of GETs. |  |
| 11 | SOAP is prone to “array overflow” attacks (similar to a buffer overflow) where a CML request with an overlarge array length is specified to a service provider (resulting in a work space for illicit code execution). |  |  |
| 12 | SOAP headers contain information which can be altered in transit (man in the middle) to cause harm (e.g. to point to an invalid file reference). |  |  |
| 13 | Security scanners (like an IPS or a Web App Firewall) are needed for comprehensive SOAP/XML message analysis. However, these can be placed in-line with XML accelerators. | "Common" IDS/IPS systems tend to be sufficient for scanning of REST (HTTP) traffic. An XML security scanner may not have any effect on REST traffic (i.e. unless XML is embedded in messages, which is not common for invocation). |  |
| 14 | SOAP messages include specification of the data type for each of their arguments (e.g. string vs. arrayofstring). This makes some types of attack (e.g. SQL injection) more difficult to initiate. |  |  |

WSDL breaks down web services into three specific, identifiable elements that can be combined or reused once defined.

The three major elements of WSDL that can be defined separately are:

* Types
* Operations
* Binding

A WSDL document has various elements, but they are contained within these three main elements, which can be developed as separate documents and then they can be combined or reused to form complete WSDL files.

## WSDL Elements

A WSDL document contains the following elements:

* **Definition** : It is the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.
* **Data types** : The data types to be used in the messages are in the form of XML schemas.
* **Message** : It is an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation.
* **Operation** : It is the abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message.
* **Port type** : It is an abstract set of operations mapped to one or more end-points, defining the collection of operations for a binding; the collection of operations, as it is abstract, can be mapped to multiple transports through various bindings.
* **Binding** : It is the concrete protocol and data formats for the operations and messages defined for a particular port type.
* **Port** : It is a combination of a binding and a network address, providing the target address of the service communication.
* **Service** : It is a collection of related end-points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.

In addition to these major elements, the WSDL specification also defines the following utility elements:

* **Documentation:** This element is used to provide human-readable documentation and can be included inside any other WSDL element.
* **Import** : This element is used to import other WSDL documents or XML Schemas.

**NOTE:** WSDL parts are usually generated automatically using web services-aware tools.

## The WSDL Document Structure

The main structure of a WSDL document looks like this:

<definitions>

<types>

definition of types........

</types>

<message>

definition of a message....

</message>

<portType>

<operation>

definition of a operation.......

</operation>

</portType>

<binding>

definition of a binding....

</binding>

<service>

definition of a service....

</service>

</definitions>

A WSDL document can also contain other elements, like extension elements and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.