SOAP (Simple Object Access Protocol)

Its a message protocol that enables the distributed elements of an application to communicate (позволяет распределенным элементам приложения обмениваться данными).or in other words is a definition of how web services talk to each other or talk to client applications that invoke them.

In today’s world, there is huge number of applications which are built on different programming languages. Exchanging data between applications is crucial in today’s networked world. But data exchange between these applications would be complex. So will be the complexity of the code to accomplish this data exchange. One of the methods used to combat this complexity is to use XML (Extensible Markup Language) as the intermediate language for exchanging data between applications.Every programming language can understand the XML markup language. Hence, (следовательно) XML was used as the underlying medium for data exchange (базовой среды для обмена данными).SOAP was designed to work with XML over HTTP and have some sort of specification which could be used across all applications.

But there are no standard specifications on use of XML across all programming languages for data exchange. That is where SOAP software comes in.

SOAP can be carried over a variety of standard protocols, including the web-related Hypertext Transfer Protocol (HTTP).SOAP was developed as an intermediate language for applications that have different programming languages, enabling these applications to communicate with each other over the internet.

Simple Object Access Protocol, as a specification, defines SOAP messages that are sent to web services and client applications. SOAP messages are XML documents that are comprised (состоит) of the following three basic building blocks:

1)-The SOAP Envelope encapsulates all the data in a message and identifies the XML document as a SOAP message.(в общем это и есть само сообщение, или контейнер в котором и расположено собственно говоря всё сообщение), The SOAP envelope indicates the start and the end of the message so that the receiver knows when an entire message has been received. The SOAP envelope solves the problem of knowing when you are done receiving a message and are ready to process it. The SOAP envelope is therefore basically a packaging mechanism.

-Every SOAP message has a root Envelope element.

-Envelope is a mandatory part of SOAP message.

-Every Envelope element must contain exactly one Body element.

-If an Envelope contains a Header element, it must contain no more than one, and it must appear as the first child of the Envelope, before the Body.

-The envelope changes when SOAP versions change.

-The SOAP envelope is specified using the ENV namespace prefix and the Envelope element.

-The optional SOAP encoding is also specified using a namespace name and the optional encodingStyle element, which could also point to an encoding style other than the SOAP one.

2)-The Header element contains additional information about the SOAP message. This information could be authentication credentials (учетные данные при аутентификации), for example, which are used by the calling application, It can also contain the definition of complex types which could be used in the SOAP message. By default, the SOAP message can contain parameters which could be of simple types such as strings and numbers, but can also be a complex object type. The optional Header element offers a flexible framework for specifying additional application-level requirements. For example, the Header element can be used to specify a digital signature for password-protected services. Likewise, it can be used to specify an account number for pay-per-use SOAP services.

**A SOAP Header can have the following two attributes −**

A) - **Actor** attribute

The SOAP protocol defines a message path as a list of SOAP service nodes. Each of these intermediate nodes can perform some processing and then forward the message to the next node in the chain. By setting the Actor attribute, the client can specify the recipient of the SOAP header.

B) - **MustUnderstand** attribute

It indicates whether a Header element is optional or mandatory. If set to true, the recipient must understand and process the Header attribute according to its defined semantics, or return a fault.

**The following example shows how to use a Header in a SOAP message.**

**<?xml version = "1.0"?> <!--**

**<SOAP-ENV:Envelope**

**xmlns:SOAP-ENV = " http://www.w3.org/2001/12/soap-envelope"**

**SOAP-ENV:encodingStyle = " http://www.w3.org/2001/12/soap-encoding">**

**<SOAP-ENV:Header>**

**<t:Transaction**

**xmlns:t = "http://www.tutorialspoint.com/transaction/"**

**SOAP-ENV:mustUnderstand = "true">5**

**</t:Transaction>**

**</SOAP-ENV:Header>**

**...**

**...**

**</SOAP-ENV:Envelope>**

3)-The Body element includes the details of the actual message that need to be sent from the web service to the calling application. This data includes call and response information. The SOAP body is a mandatory element that contains the application-defined XML data being exchanged in the SOAP message. The body must be contained within the envelope and must follow any headers that might be defined for the message.

The body is defined as a child element of the envelope, and the semantics for the body are defined in the associated SOAP schema.

The body contains mandatory information intended for the ultimate receiver of the message. For example −

<?xml version = "1.0"?>

<SOAP-ENV:Envelope>

........

<SOAP-ENV:Body>

<m:GetQuotation xmlns:m = "http://www.tp.com/Quotation">

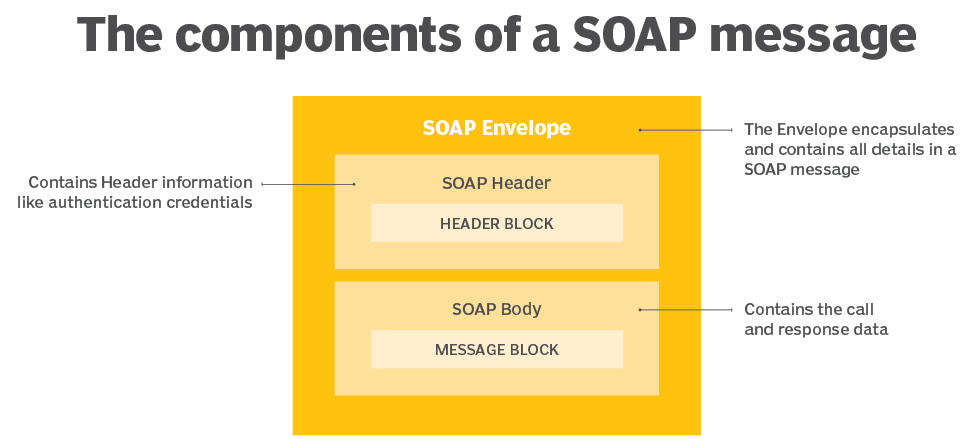
<m:Item>Computers</m:Item>

</m:GetQuotation>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

The example above requests a quotation of computer sets.(предложение комплектов компьютеров) Note that the m:GetQuotation and the Item elements above are application-specific elements. They are not a part of the SOAP standard.



**How does SOAP work?**

First, a request for a service is generated by a client using an XML document. Next, a SOAP client sends the XML document to a SOAP server. When the server receives the SOAP message, it sends the message as a service invocation to the requested server-side application (сервер перенаправляет сообщение запрошенному приложению на стороне сервера). A response containing the requested parameters, return values and data for the client is returned first to the SOAP request handler and then to the requesting client (возвращается сначала обработчику запроса SOAP, а затем запрашивающему клиенту). Both SOAP requests and responses are transported using Hypertext Transfer Protocol Secure (HTTPS) or a similar protocol like HTTP.

**the general structure of a SOAP message**

<?xml version = "1.0"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"

SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

<SOAP-ENV:Header>

...

...

</SOAP-ENV:Header>

<SOAP-ENV:Body>

...

...

<SOAP-ENV:Fault>

...

...

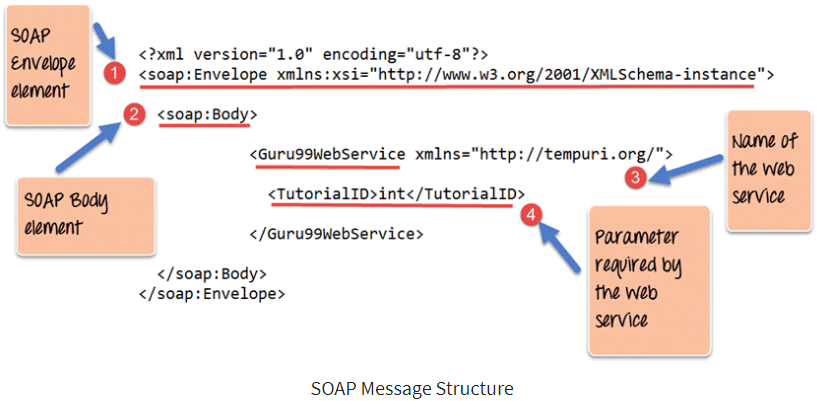
</SOAP-ENV:Fault>

...

</SOAP-ENV:Body>

</SOAP\_ENV:Envelope>

simple SOAP message structure:



1) the first part of the SOAP message is the envelope element which is used to encapsulate the entire SOAP message.

2) The next element is the SOAP body which contains the details of the actual message.

3) Our message contains a web service which has the name of “Guru99WebService”.

4) The “Guru99Webservice” accepts a parameter of the type ‘int’ and has the name of TutorialID.

**ещё Пример SOAP message:**

Content-Type: application/soap+xml

<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">

<env:Header>

<m:GetLastTradePrice xmlns:m="Some-URI" />

</env:Header>

<env:Body>

<symbol xmlns:p="Some-URI" >DIS</symbol>

</env:Body>

</env:Envelope>

—---------------------------------------------------------------------------------------------------------------------------------------------------------

SOAP schema.?

Normally, the application also defines a schema to contain semantics associated with the request and response elements.

**—-----------------------------------------------------------------------------------------------------------------------------------------------------------**

SOAP with HTTP POST

The following example illustrates the use of a SOAP message within an HTTP POST operation, which sends the message to the server. It shows the namespaces for the envelope schema definition and for the schema definition of the encoding rules. The **OrderEntry** reference in the HTTP header is the name of the program to be invoked at the tutorialspoint.com website.

POST /**OrderEntry** HTTP/1.1

Host: www.tutorialspoint.com

Content-Type: application/soap; charset = "utf-8"

Content-Length: nnnn <!--всё, что с верху это HTTP bindings (привязки)

<?xml version = "1.0"?>

<SOAP-ENV:Envelope

xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"

SOAP-ENV:encodingStyle = " http://www.w3.org/2001/12/soap-encoding">

...

Message information goes here

...

</SOAP-ENV:Envelope>

!NOTE − The HTTP binding specifies the location of the service.

—-----------------------------------------------------------------------------------------------------------------------------------------------------------

**SOAP - Fault**

If an error occurs during processing, the response to a SOAP message is a SOAP fault element in the body of the message, and the fault is returned to the sender of the SOAP message.

The SOAP fault mechanism returns specific information about the error, including a predefined code, a description, and the address of the SOAP processor that generated the fault.

Points to Note about fault:

- A SOAP message can carry only one fault block.

- Fault is an optional part of a SOAP message.

- For HTTP binding, a successful response is linked to the 200 to 299 range of status codes.

- SOAP Fault is linked to the 500 to 599 range of status codes.

**Sub-elements of Fault**

The SOAP Fault has the following sub elements −

1)- <faultCode> It is a text code used to indicate a class of errors. See the next Table for a listing of predefined fault codes.

2)- <faultString> It is a text message explaining the error.

3)- <faultActor> It is a text string indicating who caused the fault. It is useful if the SOAP message travels through several nodes in the SOAP message path, and the client needs to know which node caused the error. A node that does not act as the ultimate destination must include a faultActor element.

4)- <detail> It is an element used to carry application-specific error messages. The detail element can contain child elements called detail entries.

SOAP Fault Codes

The faultCode values defined below must be used in the faultcode element while describing faults.

1)- SOAP-ENV:VersionMismatch.

Found an invalid namespace for the SOAP Envelope element.

2)- SOAP-ENV:MustUnderstand

An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood.

3)-SOAP-ENV:Client

The message was incorrectly formed or contained incorrect information.

4)- SOAP-ENV:Server

There was a problem with the server, so the message could not proceed.

SOAP Fault Example

The following code is a sample Fault. The client has requested a method named ValidateCreditCard, but the service does not support such a method. This represents a client request error, and the server returns the following SOAP response −

<?xml version = '1.0' encoding = 'UTF-8'?>

<SOAP-ENV:Envelope

xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"

xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"

xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

<SOAP-ENV:Body>

<SOAP-ENV:Fault>

<faultcode xsi:type = "xsd:string">SOAP-ENV:Client</faultcode> ← **смотрим чё за код ошибки**

<faultstring xsi:type = "xsd:string">

Failed to locate method (ValidateCreditCard) in class (examplesCreditCard) at

/usr/local/ActivePerl-5.6/lib/site\_perl/5.6.0/SOAP/Lite.pm line 1555.

</faultstring>

</SOAP-ENV:Fault>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

**SOAP - Encoding**

SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

+SOAP data types are divided into two broad categories − scalar types and compound types.

+Scalar types contain exactly one value such as a last name, price, or product description.

+Compound (составные) types contain multiple values such as a purchase order or a list of stock quotes.

+Compound types are further subdivided into arrays and structs.

+The encoding style for a SOAP message is set via the SOAP-ENV:encodingStyle attribute.

+Latest SOAP specification adopts all the built-in types defined by XML Schema. Still, SOAP maintains its own convention for defining constructs not standardized by XML Schema, such as arrays and references.

HTTP remains the most popular SOAP transport protocol.

Quite logically, SOAP requests are sent via an HTTP request and SOAP responses are returned within the content of the HTTP response. While SOAP requests can be sent via an HTTP GET, the specification includes details on HTTP POST only.

Additionally, both HTTP requests and responses are required to set their content type to **text/xml**.

The SOAP specification mandates that the client must provide a SOAPAction header, but the actual value of the SOAPAction header is dependent on the SOAP server implementation.

For example, to access the AltaVista BabelFish Translation service, hosted by XMethods, you must specify the following as a SOAPAction header.

Even if the server does not require a full SOAPAction header, the client must specify an empty string ("") or a null value. For example −

SOAPAction: ""

SOAPAction: