Apex Basic & DataBase

Apex SOAP

creds  
leshii85@resilient-bear-bhomqg.com

xiezuB6j!@

jR4AlCoDEIz1MWdYeU8FrcvH

<https://www.youtube.com/watch?v=Tiqnt-2U3r0>

https://www.apexhours.com/integrating-with-salesforce-part-2/

https://aws.amazon.com/compare/the-difference-between-soap-rest/

https://www.apexhours.com/salesforce-integration/

1 Theory  
 - SOAP  
 - WSDL  
 - Enterprise WSDL  
 - Apex WSDL  
2 Enterprise WSDL

3 APEX soap

**SOAP**

**Meaning/Uses** Simple Object Access Protocol, uses WSDL

**Design** Standardized protocol with pre-defined rules to follow.

**Approach** Function-driven (data available as services, e.g.: “getUser”)

**Statefulness** Stateless by default, but it’s possible to make a SOAP API stateful.

**Caching** API calls cannot be cached.

**Security** WS-Security with SSL support. Built-in ACID compliance.

**Performance** Requires more bandwidth and computing power.

**Message format** Only XML.

**Transfer protocol(s)** HTTP, SMTP, UDP, and others.

**Recommended for**  Enterprise apps, high-security apps, distributed environment, financial services, payment gateways, telecommunication services.

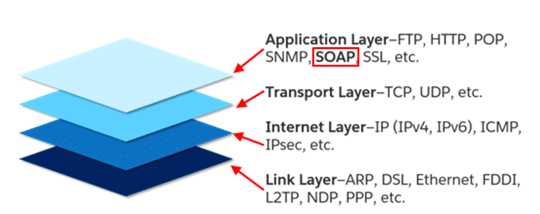
**Advantages** High security, standardized, extensibility.

**Disadvantages** Poorer performance, more complexity, less flexibility.

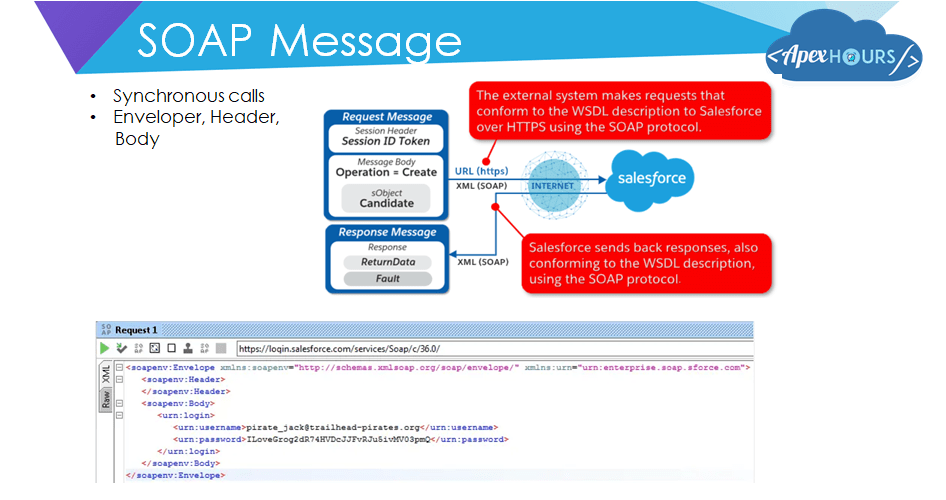
**Error Handling** SOAP response will contain error information

Enterprise – 1000 calls / license (1M call if 1000 licenses)

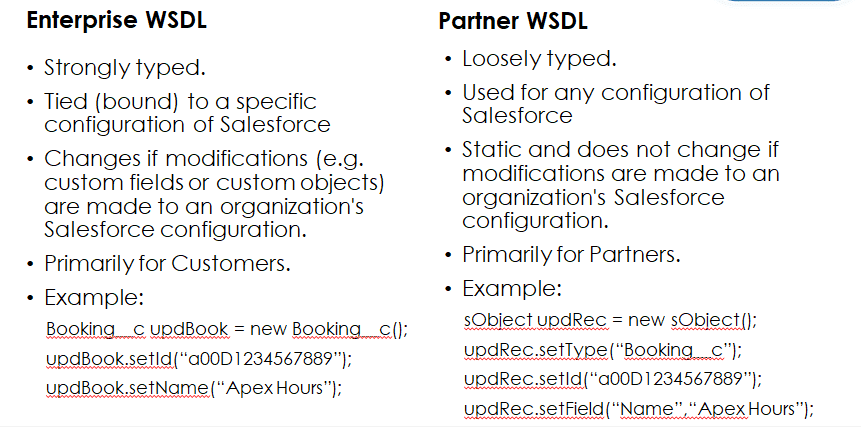
* Unlimited – 5000 calls/ license
* Application layer protocol used to exchange structured information between systems
* It uses a Web Services Description Language (WSDL) file to rigorously define the parameters for accessing data through the API.
* SOAP API supports XML only.
* Because SOAP API uses the WSDL file as a formal contract between the API and consumer, it’s great for writing server-to-server integrations.
* Access to Salesforce data and business logic
* Handles medium data volumes
* Updates multiple records with single



#### SOAP Message



#### Enterprise vs Partner WSDL



Practice  
  
**1 Enterprise WSDL**  
a) → SF Setup → API → Enterprise WSDL  
b) Open SOAP UI → create project SOAP → use Ent WSDL  
c) find ‘login’ method → clean Header and add credential username/pswrd for SF  
d) from response take sessionId (authToken) and serverUrl  
e) find ‘create’ method – add sessionId to header  
 and (use this xml to create Case object)  
 *<urn:create>*

*<urn:sObjects xsi:type="urnl:Case" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">*

*<Origin>Web</Origin>*

*<Subject>SOAP API Demo</Subject>*

*<Description>SOAP UI Project</Description>*

*</urn:sObjects>*

*</urn:create>*

**2 APEX WSDL** (when Enterprise is not enough)  
a) create apex webservice Apex class  
*global class SimpleSOAPAPI {*

*webservice static Id createCase(String origin, String subject, String description){*

*Case thisCase = new Case(*

*Origin = origin,*

*Subject = subject,*

*Description=description);*

*insert thisCase;*

*return thisCase.Id;*

*}*

*}*b) create WSDL from Apex class  
Setup→ Apex Classes → open our class and press create WSDL  
c) add this WSDl as new SOAP project to SoapUI – and call createCase method (put session id in header)  
  
d) on another instance of SF open  
Setup→ Apex Classes → create Class from WSDL button  
choose our wsdl file and leave all other as is

create RemoteSiteSetting and add to it our endpoint (allow to call this server)  
  
open DevConsole and run this code **change sessionId from auth in SoapUI**  
  
*soapSforceComSchemasClassSimplesoap.SimpleSOAPAPI soapapi = new soapSforceComSchemasClassSimplesoap.SimpleSOAPAPI();*

*soapapi.SessionHeader = new soapSforceComSchemasClassSimplesoap.SessionHeader\_element();*

*soapapi.SessionHeader.sessionId = '00DQy000007zdx1!AQEAQGhM9O0vm.Fm9m1vXGc6JS2qSUifUa17\_2QDgplF1.4qt1vdm6BfXmDitW\_Fcv6p\_CbiZP2ANDdLMRWLPuV03LmyyzVD';*

*String externalCaseId = soapapi.createCase('Email','From another SF instance','xxx3');*

*System.debug(externalCaseId);*

**Considerations for Using the webservice Keyword**

When using the webservice keyword, keep the following considerations in mind:

* Use the webservice keyword to define top-level methods and outer class methods. You **can’t** use the webservice keyword to define a class or an inner class method.
* You cannot use the webservice keyword to define an interface, or to define an interface's methods and variables.
* System-defined enums cannot be used in Web service methods.
* You cannot use the webservice keyword in a trigger.
* All classes that contain methods defined with the webservice keyword must be declared as global. If a method or inner class is declared as global, the outer, top-level class must also be defined as global.
* Methods defined with the webservice keyword are inherently global. Any Apex code that has access to the class can use these methods. You can consider the webservice keyword as a type of access modifier that enables more access than global.
* Define any method that uses the webservice keyword as static.
* You cannot deprecate webservice methods or variables in managed package code.
* Because there are no SOAP analogs for certain Apex elements, methods defined with the webservice keyword cannot take the following elements as parameters. While these elements can be used within the method, they also cannot be marked as return values.

**Maps**

**Sets**

**Pattern objects**

**Matcher objects**

**Exception objects**

* Use the webservice keyword with any member variables that you want to expose as part of a Web service. Do not mark these member variables as static.

Considerations for calling Apex SOAP Web service methods:

* Salesforce denies access to Web service and executeanonymous requests from an AppExchange package that has Restricted access.
* Apex classes and triggers saved (compiled) using API version 15.0 and higher produce a runtime error if you assign a String value that is too long for the field.
* If a login call is made from the API for a user with an expired or temporary password, subsequent API calls to custom Apex SOAP Web service methods aren't supported and result in the INVALID\_OPERATION\_WITH\_EXPIRED\_PASSWORD error. Reset the user's password and make a call with an unexpired password to be able to call Apex Web service methods.  
    
    
    
  **DOM and HttpCall for SOAP services**

String SE\_NAMESPACE = 'http://schemas.xmlsoap.org/soap/envelope/';

String SE\_PREFIX = 'soapenv';

String TE\_NAMESPACE = 'urn:enterprise.soap.sforce.com';

String TE\_PREFIX = 'urn';

Dom.Document doc = new Dom.Document();

Dom.XMLNode envelope = doc.createRootElement('Envelope', SE\_NAMESPACE, SE\_PREFIX);

Dom.XMLNode header = envelope.addChildElement('Header', SE\_NAMESPACE, SE\_PREFIX);

header.addChildElement('organizationId', TE\_NAMESPACE, TE\_PREFIX).addTextNode('00DQy000007zdx1');

Dom.XMLNode login = envelope.addChildElement('Body', SE\_NAMESPACE, SE\_PREFIX)

.addChildElement('login', TE\_NAMESPACE, TE\_PREFIX);

login.addChildElement('username', TE\_NAMESPACE, TE\_PREFIX)

.addTextNode('leshii85@resilient-bear-bhomqg.com');

login.addChildElement('password', TE\_NAMESPACE, TE\_PREFIX)

.addTextNode('xiezuB6j!@13kGtjOq1LDndoJFEW1g1UqNWh');

HttpRequest request = new HttpRequest();

//request.setEndPoint('https://login.salesforce.com/services/Soap/c/63.0/');

request.setEndPoint('callout:soaplogin/services/Soap/c/63.0/');

request.setMethod('POST');

request.setBody(doc.toXmlString());

request.setHeader('Content-Type', 'text/xml; charset=UTF-8');

request.setHeader('SOAPAction','test');

Http h = new Http();

HttpResponse response = h.send(request);

if (response.getStatusCode() != 200) {

System.debug('Response not 200 '

+ ' statusCode='+ response.getStatusCode()

+ ' status=' + response.getStatus());

}

System.debug('response=' + response.getBody());