

JangoMail Tutorial

The JangoMail Application Programming Interface (API): Using the JangoMail Web Service

Overview:

The JangoMail API is a remotely accessible web service that can be interacted with through several methods, all of which are available across the Internet. It allows you to get data into and out of your JangoMail account programmatically and without visiting www.jangomail.com.

What data can I access using the JangoMail Web Service?

The JangoMail Web Service allows you to:

- Manage bounces and unsubscribes
- Manage Groups and the e-mail addresses contained within them
- Trigger mass e-mails
- Retrieve reporting data
- And much, much more!

All of functions can be accessed programmatically from a variety of remote platforms and through a variety of methods. A full listing of methods can be found at <https://api.jangomail.com>.

Why would I use the web service?

You would use this feature if you need to remotely access data in your JangoMail account using an application running on your own desktop or server.

Here are some scenarios where the JangoMail Web Service could streamline and automate daily business tasks:

- A customer indicates on your website that they wish to unsubscribe from future mailings - your website can add the specified e-mail address to your JangoMail unsubscribe list immediately and automatically.
- On the first of the month you want to trigger a mass e-mail to distribute last month's sales activity to your sales force.
- A customer registers on your website and his information is stored into your company's CRM system - your website can also automatically add the new user and his information to a JangoMail Group.

How can I access the JangoMail Web Service?

The JangoMail Web Service is accessible via HTTP GET, HTTP POST, and Simple Object Access Protocol (SOAP) at <http://api.jangomail.com> or securely at <https://api.jangomail.com>. For a complete list of methods and their parameters see the class reference below.

How do I format a request to the JangoMail Web Service?

Accessing via HTTP GET:

Methods can be accessed by performing an HTTP GET request to a URL in this format: `http://api.jangomail.com/api.asmx/[MethodName]?Username=[Username]&Password=[Password] &[Additional_Parameter1]=[Value1]&[Additional_Parameter2]=[Value2]&...`

where `MethodName` is the function you want to perform, `Username` is your JangoMail account username, `Password` is your JangoMail account password, and `Additional_Parameter1/Value1`, `Additional_Parameter2/Value2`, etc. are additional parameters required by the method.

For example, the `AddBounce` method requires an e-mail address as an additional parameter. GET requests can be performed using your web browser or through any platform that can send an HTTP GET request such as ASP, PERL or Java.

Responses from an HTTP GET or HTTP Post are in the following format:

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://api.jangomail.com/">[Response_Code]
[Response_Description]
[Additional_Response_Line1]
[Additional_Response_Line2]
[...]
[Additional_Response_LineN]</string>
```

`Response_Code` is the status code, `Response_Description` is the descriptive name for this response code, and `Additional_Response_Line1`, `Additional_Response_Line2`, etc. are new line delimited additional text values. For example, a call to the method `GetUnsubsubscribeList` returns a response in this format:

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://api.jangomail.com/">0
SUCCESS
jeff@yahoo.com
mark@mycompany.com</string>
```

Code example in ASP/VBScript to add an e-mail address to your unsubscribe list via HTTP GET:

```
dim xml, url, RetVal, BaseURL, CommandName, Username, Password
Set xml = Server.CreateObject("Microsoft.XMLHTTP")
BaseURL = "http://api.jangomail.com/api.asmx/"
Username = "Stephen"
Password = "AlleyCat"
CommandName = "AddUnsubscribe"
url = BaseURL & CommandName
```

```

url = url & "?Username=" & Server.URLEncode(Username)
url = url & "&Password=" & server.URLEncode>Password)
url = url & "&EmailAddress=" & server.URLEncode("robert@hotmail.com")
xml.Open "GET", url, False
xml.Send
RetVal = xml.ResponseText
Set xml = nothing
'At this point RetVal is the response from the webservice
if left(RetVal, 5) = "<?xml" then 'Successful call
    RetVal = mid(RetVal, instr(instr(RetVal, "<string"), RetVal, ">")+1)
    RetVal = left(RetVal, len(RetVal) - 9) 'chop off trailer </string>
else 'errored
    Err.Raise 3, "JMSSClient", left(RetVal, instr(RetVal, vbcrLf))
end if

```

Same example in PERL:

```

require LWP::UserAgent;

$ua = LWP::UserAgent->new;
$MethodName = "AddUnsubscribe";
$Username = "demo";
$Password = "demo123";
$EmailAddress = "robert@hotmail.com";
$request = HTTP::Request->new(GET =>
"http://api.jangomail.com/api.asmx/$MethodName?Username=$Username&Password
=$Password&EmailAddress=$EmailAddress");
$response = $ua->request($request);
print $response->content;

```

Accessing via HTTP POST:

Methods can be accessed by performing an HTTP POST request to a URL in this format: [http://api.jangomail.com/api.asmx/\[MethodName\]](http://api.jangomail.com/api.asmx/[MethodName]) and posting method parameters in this format:

Username=[Username]&Password=[Password] &[Additional_Parameter1]=[Value1]& &[Additional_Parameter2]=[Value2]&...

where MethodName is the function you want to perform, Username is your JangoMail account username, Password is your JangoMail account password, and Additional_Parameter1/Value1, Additional_Parameter2/Value2, etc. are additional parameters required by the method.

For example, the AddBounce method requires an e-mail address as an additional parameter. POST requests can be performed using your web browser or through any platform that can send an HTTP POST request such as ASP, PERL or Java.

Responses to a POST request are in identical format to the response from a GET request.

Code example in ASP/VBScript to add an e-mail address to your unsubscribe list via HTTP POST:

```

dim xml, url,RetVal,BaseURL,CommandName,Username,Password
Set xml = Server.CreateObject("Microsoft.XMLHTTP")
BaseURL = "http://api.jangomail.com/api.asmx/"
Username = "Stephen"
Password = "AlleyCat"
CommandName = "AddUnsubscribe"
url = BaseURL & CommandName

params = params & "Username=" & Server.URLEncode(Username)
params = params & "&Password=" & server.URLEncode>Password)
params = params & "&EmailAddress=" &
server.URLEncode("robert@hotmail.com")
xml.Open "POST", url, False
xml.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
xml.Send params
RetVal = xml.ResponseText
Set xml = nothing
'At this point RetVal is the response from the web service
if left(RetVal, 5) = "<?xml" then 'Successful call
    RetVal = mid(RetVal, instr(instr(RetVal, "<string"), RetVal, ">")+1)
    RetVal = left(RetVal, len(RetVal) - 9) 'Chop off trailer </string>
else 'Errored
    Err.Raise 3, "JangoMailWebService", left(RetVal, instr(RetVal,
vbCrLf))
end if

```

Same example in PERL:

```

require LWP::UserAgent;

$ua = LWP::UserAgent->new;
$MethodName = "AddUnsubscribe";
$Username = "Stephen";
$Password = "AlleyCat";
$EmailAddress = "stephen@hotmail.com";

my $request = HTTP::Request->new(POST =>
"http://api.jangomail.com/api.asmx/$MethodName");
$request->content_type("application/x-www-form-urlencoded");
$request-
>content("Username=$Username&Password=$Password&EmailAddress=$EmailAddress
");

$response = $ua->request($request);
print $response->content;

```

Accessing via Simple Object Access Protocol:

The JangoMail Web Service is also fully compliant with the Simple Object Access Protocol as described by the World Wide Web Consortium (W3C) and is the preferred method for accessing JangoMail data. SOAP is a simple XML based protocol to let applications exchange information over HTTP. The main advantages of SOAP compared to other standards are:

- SOAP is designed to communicate via Internet
- SOAP is platform independent
- SOAP is language independent
- SOAP is based on XML
- SOAP is simple and extensible
- SOAP allows you to get around firewalls
- SOAP will be developed as a W3C standard

For more information on SOAP please refer to:

- http://www.w3schools.com/soap/soap_intro.asp
- <http://www.cs.fsu.edu/~engelen/soap.html> (gSOAP, C++ toolkit)
- <http://www.perl.com/pub/a/2001/01/soap.html> (Perl SOAP module)
- <http://www.w3.org/TR/soap/> (W3C specification)

The JangoMail Web Service can be accessed at <http://api.jangomail.com/api.asmx> or securely at <https://api.jangomail.com/api.asmx>.

If you are using Visual Studio .NET, the associated classes for calling the SOAP interface can be generated manually using wsdl.exe with the following call:

```
wsdl.exe /language:cs /out:JangoMailServiceProxy.csc
http://api.jangomail.com/api.asmx?WSDL
```

This will create JangoMailServiceProxy.csc which you can include in a Visual Studio project, or you can generate a DLL to use with an ASP.NET application with the following command:

```
csc.exe /t:library /out:JangoMailServiceProxy.dll
JangoMailServiceProxy.csc
```

This will compile JangoMailServiceProxy.csc to JangoMailServiceProxy.dll which you can reference in a Visual Studio project, or place in your webserver's /bin directory for use with an ASP.NET application. Sample call from a C# application:

```
JangoMail j = new JangoMail();
try
{
    j.AddUnsubscribe("Stephen", "AlleyCat", "stephen@hotmail.com");
}
catch(Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

SOAP requests can also be sent manually if formatted correctly and the correct HTTP headers are sent with the request. A SOAP request to the AddUnsubscribe method has the following format:

```
POST /api.asmx HTTP/1.1
Host: api.jangomail.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://api.jangomail.com/AddUnsubscribe"
```

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AddUnsubscribe xmlns="http://api.jangomail.com/">
      <Username>string</Username>
      <Password>string</Password>
      <EmailAddress>string</EmailAddress>
    </AddUnsubscribe>
  </soap:Body>
</soap:Envelope>

```

Here is a code sample in classic ASP that can create and submit a SOAP request:

```

BaseUrl = "http://api.jangomail.com/api.asmx"
SoapAction = "http://api.jangomail.com/AddUnsubscribe"
Username = "Stephen"
Password = "AlleyCat"
EmailAddress = "stephen@hotmail.com"

Req = "<?xml version="1.0" encoding="utf-8"?>" & vbCrLf
Req = Req & "<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">" & vbCrLf

Req = Req & "<soap:Body>" & vbCrLf
Req = Req & "<AddUnsubscribe xmlns="http://api.jangomail.com/">" &
vbCrLf

Req = Req & "<Username>" & Username & "</Username>" & vbCrLf
Req = Req & "<Password>" & Password & "</Password>" & vbCrLf
Req = Req & "<EmailAddress>" & EmailAddress & "</EmailAddress>" & vbCrLf
Req = Req & "</AddUnsubscribe>" & vbCrLf
Req = Req & "</soap:Body>" & vbCrLf
Req = Req & "</soap:Envelope>" & vbCrLf

dim xml, url,RetVal
Set xml = Server.CreateObject("Microsoft.XMLHTTP")
xml.Open "POST", BaseUrl, False
xml.setRequestHeader "Content-Type", "text/xml; charset=utf-8"
xml.setRequestHeader "SOAPAction", SoapAction
xml.Send Req
RetVal = xml.ResponseText
Set xml = nothing

response.Write(RetVal)

```

JangoMail Web Service Function Reference

For a complete reference to the web service's methods and signatures, please see http://www.jangomail.com/documents/Public/JangoMail_Web_Service_Help.chm.

JangoMail Web Service Exception Handling

If a particular method fails, the JangoMail Web Service will throw an exception instead of returning the method's actual return type. For example, an exception will be thrown if invalid parameters are passed into a method, such as an invalid username/password combination. An exception might also be thrown if there is an internal system error. We recommend writing your code to catch these exceptions in case they occur. For example, in C#, this can be accomplished using try/catch blocks. Here is an example using the AddUnsubscribe method.

```
JangoMail j = new JangoMail();
string ResultString = "";

try
{
    ResultString = j.AddUnsubscribe("test", "pass", "stephen@hotmail.com");
}
catch(Exception ex)
{
    Console.WriteLine(ex.Message);
}
if (ResultString=="0\nSUCCESS")
{
    Console.WriteLine("The unsubscribe operation was successful.");
}
```

In this code sample, if the username/password combination of "test" or "pass" is invalid, then the execution of the code will be passed to the catch block, and an appropriate error will be written to the console. Similarly if the e-mail address is an invalid e-mail address, the catch block would write a friendly error message to the console. If the method is successful, then the ResultString variable is set to "0\nSUCCESS" and the code would write out the sentence "The unsubscribe operation was successful." to the console.

All methods will only return their true return type if the method executes successfully and all input parameters are valid. If a method is unable to execute successfully for any reason, an exception is thrown instead of returning the expected object type.

Here's an example where the API is called using an HTTP POST rather than SOAP in Visual Basic:

```
Dim xml, RetVal, Username, Password
Set xml = CreateObject("Microsoft.XMLHTTP")
Username = "Stephen"
Password = "AlleyCat"

params = params & "Username=" & Username
```

```
params = params & "&Password=" & Password
params = params & "&EmailAddress=" & "robert@hotmail.com"

xml.Open "POST", "http://api.jangomail.com/api.asmx/AddUnsubscribe", False
xml.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
xml.Send params
RetVal = xml.ResponseText
Set xml = Nothing

'At this point RetVal is the response from the webservice
If Left(RetVal, 5) = "<?xml" Then 'Successful call
    MsgBox (RetVal)
Else 'Errored
    MsgBox (RetVal)
End If
```

If the username/password combination is invalid, then RetVal would be set to:

Web Service Exception LoginFailedException: Username and password do not match.

If the method does execute successfully, then RetVal would be set to:

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://api.jangomail.com/">0
SUCCESS</string>
```