A Beginners Guide to RESTful SMS Services

Sending text messages & other services

Integration Guide

Version 1.8

July 2013

www.textmarketer.co.uk

Contents

Document Change Log

Date	**Version**	**Detail**
15 July 2013	1.8	Added *sms* DELETE method for deleting scheduled SMS.
01 July 2013	1.7	New resource URL and Message Box updates
16 February 2012	1.6	Added client library ref and new available resources
12 January 2012	1.5	Added reference to PHP Client and new group methods
7 April 2011	1.4	Updated with new REST group resources
1 February 2011	1.3	Rewrite to make it easier for beginners
3 September 2010	1.2	Update to reflect new services
22 March 2010	1.1	Revised resources
3 March 2010	1.0	Initial draft

Introduction

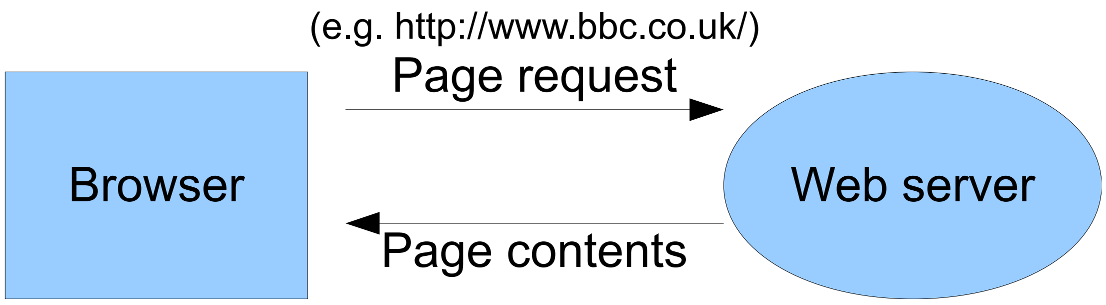
The REST API is a powerful tool for programmatically controlling your Text Marketer account. It is the most feature-rich of all our APIs and provides the most detailed information about the success/failure of your requests. Furthermore, the REST API uses standard Internet protocols, making it easily accessible from all programming languages.

For PHP users we provide a Client library, so getting started is really simple; see the document "A Quick Guide to the PHP REST Client" at www.textmarketer.co.uk.

Is there something extra you'd like the REST API to do for you? Why not get in touch and let us know about your idea.

The Basic Idea

So what's REST all about? Actually, it's just the way the Internet already works. Your web browser makes a request for a web page, and the web site responds.



However as you probably know, when you create a form in HTML, there are two method types:

- POST
- GET

You know this from HTML:

```
<form action="http://www.bbc.co.uk/dosomething" method="GET">
<input type="text" name="page" value="1" />
</form>
```

With a GET request, the form variables are sent in the URL, e.g.

```
http://www.bbc.co.uk/dosomething?page=1
```

With a POST request, the form variables are 'hidden'. They are not included in the URL but are still sent in the request:

```
http://www.bbc.co.uk/dosomething
(page=1)
```

Actually there are more than 2 methods possible for HTTP requests. Of all the methods available, REST APIs often use 4 methods:

- POST
- GET
- PUT
- DELETE

All this means is that the request type (POST, GET, PUT or DELETE) is specified in the request, in the same way as POST or GET is specified when a web browser sends form data to a web server. It's really the same process, which is partly why a REST API is said to use already-existing Internet technologies.

A REST API gives a meaning to each of these request types. The meanings are generally:

- POST – make some modification
- GET – retrieve some information
- PUT – create something
- DELETE – delete something

Therefore, they are sometimes referred to as 'verbs' (or 'actions') since they specify what to 'do':

- modify
- retrieve
- create
- delete

Do it to what?... The resource

We've seen that using a REST API we can specify an action: modify, retrieve, create or delete. So how do we specify what we want this action to be performed on?

That's where the 'resource' comes in. Actually this is just a unique URL which describes what to modify/retrieve/etc. In our case we are performing actions on a Text Marketer account... let's say we want to retrieve information about credits. This is the unique URL we use to identify our REST API *credits* resource:

```
http://api.textmarketer.co.uk/services/rest/credits
```

So the URL above specifies the *credits* resource in the Text Marketer REST API. In REST terminology it is referred to as a 'noun'. So you can see that we have 'verbs' (modify/retrieve/etc) and 'nouns' (credits, etc). This is just jargon to describe the action and the thing to do the action on.

So now we can combine the action/verb with the resource/thing/noun, for example '*retrieve* the number of *credits*' is a GET request on the credits resource URL:

```
http://api.textmarketer.co.uk/services/rest/credits (GET)
```

This returns the number of credits on the account. The response will look something like this if you view it in Firefox (Chrome only displays the number):

```
<response processed_date="2011-04-08T14:51:42+01:00">
<credits>461</credits>
</response>
```

You can try it by going to the URL http://api.textmarketer.co.uk/services/rest/credits in your web browser, and entering the **API** username and password on your account when prompted.

> INFO: This example works in your web browser because the browser automatically specifies the request as a GET request when you enter a URL.
>
> INFO: If you don't know your API username and password, you'll find them when you log on to the Text Marketer web interface here:
>
> http://messagebox.textmarketer.co.uk/#!accountsettings/ (API Config tab)
>
> For more about authentication, refer to 'Username and password – Authentication' below.

What actions can be performed?

GET requests

We've seen the GET action on the *credits* resource returns the number of credits on the account. Different actions are available on different resources. Some resources may only accept a GET action, others may only accept a POST action, and yet others may accept both.

POST requests

The *credits* resource allows the POST (modify) action. Calling a POST action on the credits resource (i.e. sending a HTTP request, of type POST, to the URL http://api.textmarketer.co.uk/services/rest/credits) allows you to transfer credits between accounts.

How does this work?

POST requests are for modifying a resource. But how do we specify exactly what needs to be modified? A POST request sent to the *credits* resource allows us to transfer credits between accounts, so we need some way of specifying how many credits to transfer and to which account.

Like an HTML form that uses a POST method, we will pass data within our POST request. In fact, if you were to build an HTML page with a form like this:

```
<form action="http://api.textmarketer.co.uk/services/rest/credits"
method="POST">
      Quantity:<input name="quantity" value="1"/><br/>
      Target account:<input name="target"/><br/>
      Target password:<input name="target_password"/><br/>
      <input type="submit" name="Test Credits POST"/>
</form>
```

...using input fields as above to specify the number of credits and the account to transfer to, you could successfully make a call to the API. However the API will respond with XML which isn't very friendly for a web browser!

So you will instead do this programmatically, within your preferred programming language. For example, this is what the same thing looks like in PHP:

```
<?php
$url = 'http://api.textmarketer.co.uk/services/rest/credits';
$data = array('quantity'=>'1', 'target'=>961,
      'target_password'=>'mypassword',
      'username'=>'u', 'password'=>'u');
$data = http_build_query($data, '', '&');
// we're using the curl library to make the request
$curlHandle = curl_init();
curl_setopt($curlHandle, CURLOPT_URL, $url);
curl_setopt($curlHandle, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curlHandle, CURLOPT_POSTFIELDS, $data);
curl_setopt($curlHandle, CURLOPT_POST, 1);
$responseBody = curl_exec($curlHandle);
$responseInfo  = curl_getinfo($curlHandle);
curl_close($curlHandle);
?>
```

If you're not familiar with PHP, it doesn't matter. All this illustrates is that we are making a call to the URL http://api.textmarketer.co.uk/services/rest/credits using a POST request ("CURLOPT_POST" in the example code), and as data we're passing:

- quantity
- target
- target_password
- username
- password

which is built into a string like this, just like you would see in a URL, before being sent to the server as POST fields:

```
quantity=1&target=961&target_password=mypassword&username=u&password=p
```

The first three arguments are what we need to specify the number of credits to transfer, and which account to transfer them to. We'll cover the username/password arguments later.

PUT requests

The *group* resource allows a PUT request, which lets you create a group. How to make a PUT request is not within the scope of this document.

DELETE requests

Currently, the only resource allowing DELETE requests on Text Marketer REST API is the *sms* resource. This is used to delete a previously scheduled SMS via API.

API capabilities – available resources

So we've seen that by specifying the type of request as GET or POST we can get the API to do different things. What other resources do we have access to and what can we do with them?

Below is a summary of the resources and the actions available on them.

Resource	**Method**	**Action**
sms	POST	Send an SMS
sms/<scheduledID>	DELETE	Delete a previously scheduled SMS
credits	GET	Get the number of credits on the account
credits	POST	Transfer a number of credits to another account
deliveryReports	GET	Get a list of delivery reports available
deliveryReport/<name>	GET (1)	Get the contents of a delivery report whose name is '<name>'. Using a name of 'all' retrieves all reports.
deliveryReport/<name>/<from>/<to>	GET	Same as 1 above, but for specific dates/times
deliveryReport/<name>/custom/<tag>	GET (2)	Same as 1 above, but restricts results to a specific custom string tag (specified when you send the SMS)
deliveryReport/<name>/custom/<tag>/<from>/<to>	GET	Same as 2 above, but for specific dates/times
keywords/<keyword>	GET	Check the availability of the keyword '<keyword>'
groups	GET	Get a list of the available send groups
group/<group name or ID>	POST	Add number(s) to a given send group
group/<group name or ID>	PUT	Create a group with the given name
group/<group name or ID>	GET	Get group details including numbers in a group

Details of the parameters (if any) required by these resources is covered in our full API documentation, available at www.textmarketer.co.uk.

NOTE: the resource names are cAsE-sensitive, e.g. 'deliveryreports' will NOT work.

Username and password – Authentication

The REST API needs to verify that you are authorised to access the account, and to identify which account you wish to access. For this purpose you must provide your API username and password. This is different to the account username and password that you use to log on to the web interface.

You can find your API username and password via the web interface:

http://messagebox.textmarketer.co.uk/#!accountsettings/ (API Config tab)

There are a couple of ways to pass this information to the API. The first is via the GET/POST variables. For example you could put them in the GET variables like this:

```
http://api.textmarketer.co.uk/services/rest/credits?username=u&password=p
```

When using the POST action, you would need to add the username/password variables to the POST arguments instead.

The second option, for those who prefer it, is to use HTTP Basic Authentication.

Response Codes and Errors

There are 2 ways of knowing whether your request was successful. The first is the HTTP status code. Just like a web server will return a 200 status code in the response headers (and HTML in the response body), when you make a successful request for a web page, so too will the REST API .

A 200 status code (returned in the response headers) indicates a successful request. This means that the resource you specified was valid, you were authenticated and the specific data you requested was found.

But you could receive any one of the following status codes in the response header:

Status Code	Meaning
200	OK. We were able to successfully execute the requested operation.
400	Bad request. There was a problem with your request data, see the status message for details.
403	Incorrect username/password, or you do not have sufficient rights to access the specified resource.
404	The resource specified could not be found.
405	The specified method (GET, POST, PUT or DELETE) is not allowed for this resource.
500	An unexpected error occurred.
501	The method requested has not been implemented.
503	Service Unavailable. The Web Service is not currently available to serve your request.

If the status code is 200, indicating a successful request, the body of the response will contain the data you requested, e.g. for the credits resource, the full body of the response will look similar to the following XML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE response PUBLIC
"-//textmarketer.biz//DTD Web Services REST 1.2//EN"
"http://api.textmarketer.co.uk/services/rest/DTD/credits_get.dtd">
<response processed_date="2011-04-12T14:00:09+01:00">
<credits>450</credits>
</response>
```

However, a status code that is not 200 means that an error was encountered. In this case the full body of the response will look something like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE response PUBLIC
"-//textmarketer.biz//DTD Web Services REST 1.2//EN"
"http://api.textmarketer.co.uk/services/rest/DTD/errors.dtd">
<response processed_date="2011-04-12T14:02:00+01:00">
<errors>
      <error code="404">Not Found (ERR255)</error>
</errors>
</response>
```

You can see that the XML above contains a list of errors - in this case containing only one entry. The error codes will be specific to the nature of the error. In this case we attempted to access a resource that doesn't exist. So here the error code is the same as the response header status code 404, meaning 'not found'.

However error codes are sometimes specific to the resource being called. For example, the *sms* resource might return errors like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE response PUBLIC
"-//textmarketer.biz//DTD Web Services REST 1.2//EN"
"http://api.textmarketer.co.uk/services/rest/DTD/errors.dtd">
<response processed_date="2011-04-12T14:12:04+01:00">
<errors>
      <error code="10">invalid number or not an integer</error>
      <error code="9">invalid number or too short</error>
      <error code="7">invalid message or missing</error>
      <error code="4">invalid originator or missing</error>
</errors>
</response>
```

In this case, no data was passed to the resource except for the username and password. The errors therefore describe problems with the data passed, like the 'message' parameter being missing.

INFO: The exact error codes are specific to the resource and are documented in the full API documentation, available at www.textmarketer.co.uk.

Security

All the REST API services are available using an SSL connection, i.e. using https URLs instead of http.

Testing with the Sandbox

A sandbox system is available to you for testing. The sandbox will not modify your account in any way, but will respond normally to your requests. In other words if you use the sandbox to get the number of credits on your account, it will provide the correct number. However if you use it to transfer credits between accounts, it will simulate a successful transfer (assuming you have enough credits on the account, and provide valid target account details, etc.) but it will not actually modify the number of credits on either account.

Similarly, the sandbox system will not actually send SMS messages and will not deduct credits for simulated sends.

To use the sandbox simply replace this part of the resource URL:

```
api.textmarketer.co.uk
```

with this:

```
sandbox.api.textmarketer.co.uk
```

e.g. the credits resource URL becomes:

```
http://sandbox.api.textmarketer.co.uk/services/rest/credits
```