# The VerticalResponse API Guide

Revision 4, 9/2012

# Table of Contents

# 1. Introduction

You already know how easy it is to create and send professional email campaigns with the VerticalResponse website (and if you don't, you should sign up for a free account now – it will help as you follow this guide). But did you know you can leverage this power and automate any of the email marketing functionality VerticalResponse provides?

**Anything you can do with email marketing on our website, you can do programmatically through the VerticalResponse API**, including:

- Creating and uploading a list,
- Adding to and editing existing lists,
- Creating, editing, and sending email campaigns, and
- Retrieving campaign statistics, including opens, clicks, unsubscribes, and bounces.

Our robust API provides opportunities for automation and integration that can save you, your marketing department, and your customers time and money. If you sell software to your customers that contains their contacts' email addresses, you can seamlessly add VerticalResponse to your system and dramatically add value for your customers by making that contact information actionable. The possibilities are virtually limitless.

## About This Guide

The purpose of this guide is to give you a complete introduction to the VerticalResponse API and to help you get started integrating your code with our email marketing functionality. Within this guide, you'll find recommendations, best practices, and ideas to help you get going and inspire your development.

The VerticalResponse API is quite extensive, so it's not possible to cover every method here. We encourage you to explore the API Documentation online for the full set of features available.

Because the API is implemented as a SOAP web service, it is completely language agnostic. Within this guide, we have not supplied any code samples since the code will be different depending on which language and which SOAP client you are using. We encourage you to download the code samples found on the Developer Network to help you get started.

## Enterprise API vs. Partner API

The VerticalResponse API comes in two flavors: the Enterprise API and the Partner API.

**If you only manage a single VerticalResponse account for yourself or your company, you can leverage the free Enterprise API to automate many of the processes you perform each day.** For example, if you find yourself regularly running a manual query on your database to produce a CSV file to upload to VerticalResponse as a new list, you can use the Enterprise

API plus a little bit of scripting code to automate the process into a single step. You can even send automated emails to new signups on a scheduled basis. To get started with the Enterprise API, contact the VerticalResponse API support team and request that the API be enabled for your account.

**If you manage several VerticalResponse accounts or intend to resell our services, the Partner API is the right solution**. In addition to all of the functionality available in the Enterprise API, the Partner API allows you access and maintain the accounts you control. Here are a few example integrations:

- If you provide a Customer Relationship Management (CRM) system to your customers, your customers can segment a list using your tools, then send it to VerticalResponse via the API as a new list.
- If you provide a Content Management System (CMS) to your customers that generates HTML, you can leverage those tools to allow your customers to create emails using the same system they use to manage their website, then send that HTML to VerticalResponse as a new campaign.
- Our Single Sign On feature allows you to seamlessly log your customers into their VerticalResponse account to complete the rest of their mailing.

To get started with the Partner API, contact the Partner sales team and they can give you all the details of that program.

## Email Campaign Creation Workflow

Creating and sending an email campaign follows a well-defined workflow. Using the API, you can automate any part of this workflow or even the entire workflow. The standard steps to launch an email campaign are:

1. Create a new list
2. Upload list members to that list
3. Create a new email campaign
4. Add and edit the content to that campaign
5. Preview the email content and send a test email to yourself
6. Attach one or more email lists to the campaign
7. Launch the campaign
8. Retrieve the email statistics

# 2. Accessing the VerticalResponse API

In order to use the **Enterprise API**, it must be enabled for your account; you can request access by filling out the API Request form. Once the API is enabled for your account, you'll be ready to integrate with VerticalResponse. **Accounts using the Enterprise API do not require client SSL certificates; only the Partner API requires certificates.**

If you would like to enable the **Partner API** for your existing account, you should contact your partner success manager. If you are not already a VerticalResponse partner, you should visit the Partner Program website and fill out the request form.

Once the Partner API is enabled for your account, you will receive a confirmation email along with two SSL client certificate files and their passphrase. These certificates provide an additional layer of authentication to ensure that only the holders of the certificates are able to access your subaccounts. Once you have installed the certificates according to your programming language and operating system, you will be able to create and access all of the subaccounts under your partnership.

# 3. Getting Started with the VerticalResponse API

## Logging In to Create a Session

Before you can make any other calls to the VerticalResponse API, you must first call the *login()* function and get a session key. This key is required for all subsequent method calls and is only valid for the time you specify with the *session_duration_minutes* parameter. You should set this parameter high enough to ensure that all functions for that session complete in the given time. A good starting value is 5 minutes, but it can be set as high as 120 minutes – a full two hours.

If you are using the **Enterprise API**, you'll need to provide the username and password for your VerticalResponse account. If you're using the **Partner API**, you must use the username and password for your master VerticalResponse account. If you intend to access one of your subaccounts via the API, you must provide the username for that account in the *impersonate_user* parameter of the *login()* call.

**When should you use impersonate_user?**
As a **Partner API** user, you will be performing a number of functions on behalf of your subaccounts through the API. You should **not** pass your customer's email address and password directly with *login()*; rather, you should log in as the master account and use *impersonate_user* with the user's email address.

When performing administrative tasks on subaccounts directly through your master account, do not use the *impersonate_user* parameter. Administrative tasks include *editUser()*, *getUserSignOnURL()*, *createUser()* and *createCompany()*. A good rule of thumb: **if a user ID, company ID or username is part of the call you're making as a partner, you should *not* use the impersonate_user parameter**.

Any functions you're performing **within a subaccount** – such as *createList()*, *createEmail()* or *appendFileToList()* – require the *impersonate_user* parameter (unless, of course, you're performing these functions within the master account).

# 4. List Creation and Management

## Creating Lists

Creating a new list is as easy as giving it a name and calling *createList()*. This will create an empty email list with the standard set of fields, to which you can add list members. The data you add to your list can then be merged into your email campaigns or used within our segmentation tools to sub-divide and create new lists based on various parameters.

Merge fields are identified by {curly braces} and |pipes| within them to provide a default in the case that the list doesn't contain any merge data. Here is an email salutation as an example:

Hello {FIRST_NAME|Subscriber},

These are the standard list fields and their corresponding merge fields:

| **Standard Field Name** | **Merge Field** |
|---|---|
| Email Address | {EMAIL_ADDRESS} |
| First Name | {FIRST_NAME} |
| Last Name | {LAST_NAME} |
| Created Date | {CREATE_DATE} |
| Title | {TITLE} |
| Company Name | {COMPANY_NAME} |
| Address 1 | {ADDRESS_1} |
| Address 2 | {ADDRESS_2} |
| City | {CITY} |
| State / Province | {STATE} |
| Postal Code | {POSTALCODE} |
| Country | {COUNTRY} |
| Work Phone | {WORK_PHONE} |
| Home Phone | {HOME_PHONE} |
| Mobile Phone | {MOBILE_PHONE} |
| Fax | {FAX} |
| Marital Status | {MARITAL_STATUS} |
| Gender | {GENDER} |

For data beyond these standard fields, VerticalResponse allows you to define about 65 unique custom fields for your lists ranging in size from 25 characters to 255 characters. To specify the custom fields for a list, you can provide them as a simple array of strings using the field names as values in the *custom_field_names* parameter. Custom fields can also be used to merge data into emails or with our segmentation tools to create new lists.

The *createList()* method will return a list ID that will be used for all subsequent calls made for that list. If you're maintaining your contacts database within your own system, you should store the list ID and the list name for easier reference through your own interface. You can also get a list of active lists within your account by calling *enumerateLists()*.

**How VerticalResponse Manages List Data**
Every VerticalResponse account has a Master List, which all other lists are derived from. For this reason, editing any data associated with a given list member will also update that data in all other lists within the account.

Although the Master List cannot be deleted, all the list members can be deleted from it, leaving it (and any sub-lists) empty.

## Adding List Members

There are two methods used for adding list members to a list. Which method to use depends on how many list members you're adding at a time.

If you're adding a single recipient to a list, use the *addListMember()* method. Each call to *addListMember()* can take several seconds to complete, so looping over it to add a large number of lists members at one time is rather inefficient.

In the case where you're adding more than one list member at a time, you should call *appendFileToList()*. This method accepts a large string, formatted as a CSV file and base-64 encoded, as a parameter.

If your application adds members to a list after they have filled in a form on your site, you may find it more efficient to store that information locally and batch upload the list one or several times a day using the *appendFileToList()* method rather than calling *addListMember()* for each signup. You can expect any call to a remote server to generate some lag. Reducing the number of calls you make will speed up your processes considerably.

## Updating and Deleting List Members

You can update individual list members using *editListMember()*; the list member is identified by a hash, which can be retrieved using the method *getListMemberByEmailAddress()*, which returns all stored list member data, including their hash.

If you need to update multiple list members, we recommend using the *appendFileToList()* call. You can set several parameters, such as *favor_existing_values*, to affect how *appendFileToList()* handles situations where an uploaded list member already exists.

You can remove list members by calling *deleteListMember()*. However, you should be aware that removing list members can affect your campaign reporting. In some instances, the VerticalResponse reporting tools use summary information to show you how many recipients opened or clicked on an email. In other instances, the reporting tools tally the campaign details, which are tied directly to the lists used in the email send. Deleting these lists or their list members will render some reporting inaccurate.

If you're removing list members because they either requested to be unsubscribed from outside the VerticalResponse system (i.e. they sent you an email directly or called you on the phone) or because you know their email address is invalid, you may prefer to simply mark them as unsubscribed in the system instead. You can either do this manually by logging in to your account through the VerticalResponse website or you can set the list member's *optin_status* using the *editListMember()* call. To mark them as unsubscribed, simply set this value to '1'.

### Retrieving Bounces and Unsubscribes

You can download the complete list of all email addresses that have either unsubscribed from your account or have been marked in our system as undeliverable by calling the *downloadCompanyUnsubscribesAndBounces()* method. This will return a URL that points to a CSV file containing a complete list of all unsubscribed email addresses and hard bounces for your account.

## 5. Email Campaigns

### Anatomy of an Email Campaign

The Email Campaign consists of every piece of information required to send an email, including:

- List IDs for the list or lists the campaign will be sent to,
- Content for both the text and HTML versions of your email,
- Preview and test versions of the email content,
- When the email should go out, and
- Statistics showing who on your lists opened, clicked, unsubscribed, or bounced.

Any email you send through VerticalResponse has all of this information associated with it. Thus, Email objects are full of information that goes beyond just the content.

**Why We Require a Text Version**
We require both HTML and text versions for all campaign types (except text-only *freeform_text* campaigns). Because some email addresses are configured to accept only plain-text emails, HTML emails are sent in a multi-part MIME format. This allows all subscribers to only receive a single version of their email, in their preferred format.

Our web interface creates a plain text version for HTML emails automatically, so if you are creating an email via the API, then editing it through our email creation tools, we will generate that text version on your behalf. If you are working exclusively via the API, you will need to include the plain text version using the *freeform_text* attribute.

## Email Type

VerticalResponse provides types of four email campaign: Wizard, Canvas, Freeform HTML, and Freeform Text. The type is set with the email_type attribute using one of the four options below.

> **Wizard** refers to the drag and drop email creation Wizard that can be found within the VerticalResponse web interface. You can access campaigns of this type and their associated statistics through the API, but they can only be created via the web interface.
>
> **Canvas** email campaigns can be created through both the API and the web interface. This type is ideal for creating an email campaign through the API that you intend to edit using our WYSIWYG (What You See Is What You Get) editor, called the Canvas editor.
>
> **Freeform HTML** campaigns (passed as *freeform_html)* allow you to create an HTML and text email through the API when you don't intend to edit it via our WYSIWYG editor.
>
> **Freeform Text** campaigns (passed as *freeform_text*) are ideal for sending plain-text emails. Note: text-only campaigns do not include open tracking, since we track via a transparent tracking GIF that pings our servers when the email is opened.
>
>> *Freeform_text* is also the attribute you will set if you are creating your email exclusively through the API. See the inset "Why We Require a Text Version" above for more information.

## Email Content

If you are sending HTML content via the API (whether to a canvas or freeform HTML email), you will set the content using the attributes *freeform_html* and *freeform_text*.

## Name

This is the name of the campaign as it will appear within the web interface as well as in the objects returned by the *enumerateEmailCampaigns()* method. This name must be unique within your account; you cannot have two campaigns with the same name.

## Subject

This is the subject line of the email. Unlike the name of the email, it can be a duplicate of a previously sent email campaign, though that's not always considered a best practice.

## From Label

When a campaign is sent to your customers, the *from_label* is the string they will typically see in the "From" line in their email client. The *from_label* must fit the content of your email, otherwise your campaign could be declined. For example, if you set the *from_label* to the sender's name, it should be clear from the content of the email, including the email footer, that the individual you named did send that email.

### Reply To Email

Some of your recipients may reply directly to an email sent through our system. If the reply is not an automated reply, bounce message, or request to be removed from your mailing list, our system will automatically forward it to the email address specified in the *reply_to_email* field.

## Creating an Email Campaign

Call *createEmail()* to create your campaign. You can launch your campaign directly through the API, or your users can log in to the VerticalResponse site to edit the campaign. If you have a Partner account, you can use Single Sign On to direct your users directly to the editing page of the draft email you just created.

If your users won't be editing the email through our website, you should set the *tested*, *previewed_html*, and *previewed_text* arguments to true in the *createEmail()* call to bypass those steps.

After you've created your campaign, you can edit it by using *setEmailCampaignAttributes()* or *updateEmailContents()*. When your email is ready to go, use the *setCampaignLists()* method to assign the lists of recipients to your email campaign.

> **Note:** While still listed in the API documentation, the deprecated *createEmailCampaign()* method creates an email campaign using the older editors, which are no longer supported. You should use the newer *createEmail()* method instead, because many related calls will not work with *createEmailCampaign()*.

## Launching an Email Campaign

You can call *launchEmailCampaign()* on any campaign still in draft mode that was created either using the *createEmail()* method or via our website. Calling *launchEmailCampaign()* will send your campaign to our checkers for final review and, if approved, mailed to the lists you have specified.

The status of a campaign will change as it passes through our approval process, which you can confirm by calling *enumerateCampaigns()* and checking the status parameter for each of your campaigns. Please note that the statuses returned by the API do not necessarily match the wording found within the website.

While in draft mode, all campaigns have a status of 'active'. Immediately after launch, the campaign will enter a status of 'pending approval'. It will remain in this state until our campaign checkers have approved the email, at which time it will enter a status of 'pending launch'. If the campaign is declined, the status will return to 'active' status. You can retrieve the decline history of a specific campaign by calling *getEmailCampaignDeclineHistory()*.

If you would like to set the email to launch at a date sometime in the future, you should either set the mail_date parameter to this future date in your call to *createEmail()*, or you can update the campaign’s mail date by calling *setEmailCampaignAttribute()* using *mail_date* as the parameter name and an ISO 8601 formatted date for the value.

# 6. Campaign Reporting and Statistics

## Summary Email Statistics

VerticalResponse automatically handles all email and URL tracking on your behalf for every email that is sent through our system. We gather and maintain the information for the four primary statistics used in email marketing: Email Opens, Link Clicks, Bounced Addresses, and Unsubscribe Requests.

### Email Opens

Each HTML email sent through our system includes a small tracking image hosted by our site that allows us to see who opened your HTML email and when.

### Link Clicks

Before sending your email campaign, our system rewrites each HTML link by replacing it with a unique tracking URL. When a recipient clicks on that link, they are briefly sent to our tracking servers before immediately being redirected to your original link. This way, we are able to see who has clicked on which links within your email and report that information back to you. Using this information in combination with the email open data shows you who is engaged with your email communications and can allow you to craft more targeted messages that improve conversions.

To create personalized URLs, you can use custom list fields and merge fields in your URL. If the customized field appears in the query string of your URL, you can even track who is clicking through to their personalized URL. For example, if the customer field in your list is named “user_id”, you can format such a URL like this:

```
http://www.example.com/profile/?id={user_id}
```

If the merge field must appear as part of the actual URL, however – either as part of the domain name or part of the URL path – you will not be able to track that link and will need to turn off tracking just for that URL. You can accomplish this by prefacing the URL with “nr_”, like this:

```
nr_http://www.example.com/profile/{user_id}
```

### Bounced Addresses

Emails that cannot be delivered to their destination are considered to have bounced. If the receiving server reports that the address doesn’t exist, is poorly formatted, the server itself is not responding or any other similar permanent failure, that is considered a “hard bounce” and is immediately reported as undeliverable.

If the receiving server reports a temporary problem delivering the email – i.e. the mail box is full – then we will continue to try sending for up to 72 hours. If the email is still undeliverable after 72 hours, it will be marked as undeliverable in our system.

#### Unsubscribe Requests

In order to ensure we maintain compliance with the CAN-SPAM act, VerticalResponse must handle all unsubscribe and opt-out requests through our system. Each email sent through our system has an unsubscribe link embedded in the footer. When a customer clicks on this link, they are immediately marked as unsubscribed in your account, and we will no longer send them any emails from your account. Only one unsubscribe link can be present in each campaign, and emails with multiple unsubscribe links are declined by our checkers so they can be edited to include only one unsubscribe link.

A summary of these four statistics can be accessed using the *getEmailCampaignStats()* method, which will return the summary data for the specified campaign.

### Detailed Campaign Reporting

To get the most value out of your email campaign statistics, you should consider implementing the *downloadCampaignRecipientResults()* method. This method provides a link to a CSV file containing detailed information on the actions taken by each list member for a given campaign. Using this information, you can discover who is most directly engaged with your email marketing.

# 7. Creating and Managing Partner Accounts

## Creating Subaccounts (Partner API)

The **Partner API** allows you to programmatically create and manage as many subaccounts as you need. Before creating any subaccounts through the API, however, you should first confirm that you have properly installed your client SSL certificate (See Appendix B). Subaccounts created without a certificate will be accessible by logging in via the VerticalResponse API, but they will not be tied to your partner account so you will not be able to access them either though the API or the Partner Dashboard.

Creating an account is a two-step process. You must first call *createCompany()* to create the Company object, which contains the contact information used throughout the VerticalResponse site and in the footer of each email that goes out. The *createCompany()* method will return a company ID that you will need for the next call: *createUser()*.

The User object created by the *createUser()* method will contain the email address that acts as the username for the subaccount. It also contains much of the same contact information as the

Company object. Even though this is duplicate information, we suggest you supply the same contact information that you supplied to the *createCompany()* method.

The *createUser()* method will return a user ID. Though you will primarily use the username to access a subaccount, you should also store the user ID and associate it with the email address for the account. A subaccount user, under some circumstances, may be able to change the email address they use to login to their account. However, their user ID will never change. In such an instance, you can retrieve the current email address for a subaccount by calling the *getUser()* method with the user ID as a parameter.

The existence of separate company and user objects may tempt you create multiple users for each company or to add the same user to multiple company objects. **Resist this temptation**. At this time, there is a one-to-one relationship between each company and user. **If you assign the same company ID to multiple user objects, the system will behave in unexpected ways.** If you have multiple accounts for a single company, simply call *createCompany()* for each account you create with the same company name and contact information, retrieving a separate and unique company ID for each user.

Similarly, the VerticalResponse system uses the email address as a unique identifier. This means that **you cannot create multiple subaccounts using the same email address as a username**. Doing so will generate a SOAP fault that indicates the username you provided is already in use. If you run into this error, you should either use a different email address or, if appropriate, contact your partner success manager and request that the existing account be added to your partnership.

## Single Sign On (SSO, Partner API)

If you are using the **Partner API**, you can use the Single Sign On (SSO) feature to more tightly integrate the VerticalResponse website with your system and keep your customers from having to provide an extra set of credentials in order to access their VerticalResponse account.

The SSO feature consists of a single call to *getUserSignonURL()* with the username of the subaccount you wish to access. This method returns a URL that you should then forward to your customer's browser, either as the *src* parameter for an iFrame tag or as a new window. This URL will establish a session with their browser and automatically log them in to their VerticalResponse account on our website.

If you supply the *post_login_path* parameter with this method, you can direct the user's browser to a specific page on the VerticalResponse site. This is especially useful if, for example, you are creating a new email campaign for the user using a template stored on your system that you would like them to edit using our email creation tools. In this case, you can provide a special URL in our system designed for this specific purpose:

http://app.verticalresponse.com/app/emails/email/edit/[CAMPAIGN_ID]

You should replace [CAMPAIGN_ID] in this URL with the ID returned from your call to *createEmail()*.

One of the side effects of SSO is that your system will be responsible for handling all account creation and login activities. The cobranded URL associated with your partnership will redirect to a URL that you provide to us, which will also be called should our system lose the session with your user. The logic at this URL should either allow them to log back in to your system, or for a better user experience, this URL should first check whether the user still has a session with your system. If so, it should automatically log them back in to their VerticalResponse account using the *getUserSignonURL()* method. Otherwise, it would present them with your login page.

Single Sign On is not enabled by default for your account. In order to have it enabled, you should contact your partner success manager and provide them with the redirect URL described above.

# 8. Appendix

## Appendix A: Support Resources

The VerticalResponse team wants to help make your integration with our API a success. We have created several resources to help you find answers to your questions.

### The Developer Network

The Developer Network is the primary resource for both the Enterprise and Partner API. All of our API documentation is kept up to date and is available online here. You can also find links to articles, guides, and other information intended to provide inspiration and answer any questions that may arise.

### API Email Support

If you're integrating with the Enterprise API and run into a challenge you can't get answered elsewhere, feel free to reach out to us via email at api-support@verticalresponse.com.

### Contact your Partner Success Manager

If you are a VerticalResponse Developer Partner, feel free to reach out to your partner success manager and the Sales Engineer with any questions you may have during your integration. The Sales Engineer can also help you and your team architect a solution to use the Partner API to add value to any of your existing services.

## Appendix B: Installing SSL Client Certificates (Partner API)

When your Partner API account is activated, you will be sent two certificates and an associated passphrase. You should only need one of these certificates, but the one you use will depend on which language you are developing with.

- If you are using **PHP**, **Ruby** or **Perl**, you will want to use the certificate with a .PEM extension.
- If you are using **Java** or **C#/.Net**, you will want to use the certificate with the .P12 extension.

How your system accesses these certificates depends on which SOAP library you use, which language you develop in, and which operating system your system is hosted on. For complete details, please download the sample code for your preferred language from the Developer Network. However, the following should be true in most situations:

### Ruby

Our Ruby sample code assumes that you will be using the soap4r-ruby1.9 gem, but you may prefer to use the Savon SOAP client gem. In either case, you only need place the certificate file in a location on your server that is readable by your code and load it directly during your initial call to instantiate the SOAP client object. The client will automatically send the certificate with each call to our server provided you use the same client object.

### PHP

Though third partly clients such as NuSOAP exist for PHP, the SoapClient object – included as an extension with the default PHP installation – should be all you need. Like Ruby, you simply need to place the certificate file in a location on your server that is readable by your code. You'll then provide this path as well as the certificate passphrase when you first instantiate the SoapClient object.

### Perl

Your best bet for accessing the VerticalResponse API with perl is to use the SOAP::Lite CPAN module. To allow SOAP::Lite to access your certificate, you'll need to set three environment variables - HTTP_CERT_FILE, HTTPS_KEY_FILE (both of these should point to your PEM file) and HTTPS_CERT_PASS. For more information, see the SOAP::Lite CPAN documentation.

### Java

In order to generate the stub classes from our WSDL, you should use the Axis 1.4 library from the Apache Foundation. The newer Axis2 libraries are incompatible with RPC-encoded WSDLs, so they cannot currently be used. In order to set up a certificate in Java, you'll need to add it to a Java trust store using the keytool program included with the Java SDK. Complete details can be found within the documentation directory included in the sample code.

### C#/.Net

The latest version of WCF for .Net 3.5 and 4.0 allow you to generate the stub code as a service reference from within Visual Studio. Simply provide the URL to the Partner API WSDL and your IDE will handle the rest. You must then add a new behavior element in the app.config that you reference in the endpoint element of the client block. A complete example of this is included in the documentation supplied with the sample code.

To install the certificate itself, you should call the certmgr.exe program from the “Run” window under your start menu and open it manually. Be sure to install the certificate in your Trusted Root Store in the Local Computer. You should mark it as exportable to be sure it’s readable by your system. On some versions of Windows Server, you may also need to supply direct access to the Network Service user using the FindPrivateKey tool.