Verify User Implementation Guide

How to implement Verify, best practices, and
examples to get started quickly

Contents

Implementation Guide Overview

This guide outlines how to implement *Etumos Verify* in your Marketo instance, and/or directly on your form pages to prevent unwanted, invalid email addresses from reaching your database.   Learn the technical components of the product, recommended best practices, and detailed instructions for implementation that meets your business needs.

User Implementation Guide Objectives
- Understand Etumos Verify product and how it works
- Implement Verify in your Marketo instance
- Implement Verify wherever you collect email addresses for real time email verification
- Learn how to prevent

Understanding Verify and How It Works

Etumos Verify is a real-time email verification product created specifically to work with Marketo's infrastructure, providing Marketo clients a seamless solution for boosting email deliverability rates, and maintaining clean email data. Verify validates if an email address is emailable, and responds with valuable details about the email address validity which can be used in your strategic marketing and sales efforts. Verify results are 100% accurate, as if you had tried to send the email yourself at that exact moment. Etumos Verify is your solution to keeping your marketing automation platform's data clean. To learn more about Verify including common use cases, pricing and more visit our Product Overview page [here](#).

Key Components

A completely configured Verify execution consists of the following key components:
- Javascript Snippet - customized and added to front-end pages with Marketo Forms 2.0
- Verify Webhook - A single GET webhook call with custom HTTP headers and response mapping
- Marketo Program with smart campaigns, flows, and trigger(s) configured to meet your business email verification needs

How it Works

Etumos Verify technology uses a trigger to validate an email address in real-time, responding with technical details about the email address validity. There are two primary configuration methods, which differ slightly in functionality, though remain consistent at the core. The email verification *on-form validation* method, uses a Javascript snippet configured on front-end pages where email addresses are collected to proactively prevent any invalid, or unwanted email addresses from form submission. Thereby proactively preventing an invalid email address from reaching your database. In a secondary configuration method, a Verify webhook is configured in Marketo to be called by a Flow step within a Smart Campaign, and Verify response data is then used to trigger additional steps and reach intended business outcomes. The webhook configuration method is often employed to scrub invalid email

addresses which already exist in your database. Both configuration methods can be employed simultaneously, or exclusively at any time.

Implement Verify on Form Pages

Prevent invalid or unwanted email addresses from form submission proactively by adding the customizable Verify JavaScript snippet to any pages where a Marketo Form 2.0 is embedded.

What You Need Before you Begin:

- Your unique Verify API key
- The Verify Embed Script (copied below)
- Provide Etumos with IP addresses/domains to whitelist
- Access to update entry point pages
- Email domains to block from form submission - Optional
- Custom error message copy - Optional

Configure Verify Javascript Snippet

Customize Script with Your Unique Credentials

Update the Javascript snippet below by replacing the script "src url" with the API Secret provided by Etumos upon signup.

```
<script src="//cdpn-js.figureone.com/vrfyx4marketo-1.28.1-bundle.js?{INSERT API KEY HERE}"></script>
```

Customize Domains to Prevent From Form Submission

Optionally, adjust the script to block/accept/or add email domains to reject, and prevent from form submission. This configuration is commonly applied to prevent personal email domains, and or competitor domains to avoid creating nonfunctional leads in their database. To customize this feature, add the applicable objects(s) outlined below to your Javascript snippet based on desired results:

- **To override the built-in-list of free email domains** prevented from form submission by default, add `window.vrfyx_gratis` and define an array of domains to override the built-in list In the example below, email addresses which end in *etumos.com* or *cow.com* are rejected.

```
<!-- Etumos Email Verify code below -->
<script>
window.vrfyx_gratis = ['etumos.com','cow.com'];
</script>
<script
src="//cdpn-js.figureone.com/vrfyx4marketo-1.28.1-bundle.js?{INSERT API KEY HERE}"></script>
<!-- /End Etumos Email Verify code -->
```

- **To add custom domains** to the existing built-in-list of free domains being rejected, add `window.vrfyx_custom` and define the custom domains within an array
- **To disable the email domain blocking feature,** use the `window.vrfyx_gratis= []` object and define an empty array.

Customize Error Messages for Invalid Emails

Optionally, adjust the script to customize error response messages and behaviors when an invalid email address is detected by adding the `window.vrfy_options` object and defining the 'message' variables using the pattern shown in the script below. The default values of the error messages are shown in the code block below. Replace the default values with your custom message using the format demonstrated with default values.

```
window.vrfy_options = {
        messages: {
            emailExtendedValidationRejectUserCustom: 'This email address is not allowed.',
            emailExtendedValidationRejectNonexistent: 'It looks like this email address doesn\'t
exist!',
            emailExtendedValidationRejectDisposable: 'Please enter your permanent address!',
            emailExtendedValidationRejectFree: 'Please enter a corporate address!',
            emailExtendedValidationError: 'A validation error occurred.'
        }
};
```

Implement Javascript Snippet on Pages using Marketo Forms 2.0

Embed the customized Javascript snippet to all pages with a Marketo Forms 2.0 or wherever you collect email addresses to enable Verify to work. The example below shows a complete Javascript snippet with custom domain1 and domain2 added to the built-in-list of free email domains to prevent from form submission.

```
<script>
window.vrfyx_custom = ['{domain1}','{domain2}'];
window.vrfy_options = {
        messages: {
            emailExtendedValidationRejectUserCustom: 'This email address is not allowed.',
            emailExtendedValidationRejectNonexistent: 'It looks like this email address doesn\'t
exist!',
            emailExtendedValidationRejectDisposable: 'Please enter your permanent address!',
            emailExtendedValidationRejectFree: 'Please enter a corporate address!',
            emailExtendedValidationError: 'A validation error occurred.'
        }
    };
</script>
<script src="//cdpn-js.figureone.com/vrfyx4marketo-1.28.1-bundle.js?{INSERT API KEY
HERE}"></script>
```

Test Implementation

To test your implementation on a web browser and go to the page where verify was installed. Type in an invalid email address to confirm Verify is working. Please note for Verify to work, you must provide Etumos with the IP addresses and/or domain names of the pages Verify will be used. If the IP address has not been added to our whitelist then Verify will not work as expected.

Implement Etumos Verify by Marketo Webhook

Configure a webhook in Marketo which takes an input email address and returns response values after validating the email address. Configure a Marketo program with smart campaigns, and flow steps designed to trigger the Verify webhook call and act based on response in alignment with your business needs.

What you'll need Before you Begin

When using Verify within Marketo, before getting started ensure that you have the following:
- Your unique Verify header with API key
- The Verify endpoint URL provided by Etumos upon signup
- Administrative Access to create webhooks

Create a New Webhook in Marketo

Configure a Verify webhook in Marketo with the unique endpoint url, custom header, and your email token for Verify to validate. The custom header and URL will be provided by Etumos upon successful signup.

1. Login to Marketo and open the Admin panel
2. Open Integration and Webhooks
3. From the "Webhooks Actions" menu "Create New webhook" - Give your new Webhook and name such as "Verify". If desired, enter a description that explains what this Verify webhook does
4. Copy and paste your Verify endpoint URL directly into the Endpoint URL field
5. Replace the API Key Placeholder in the endpoint URL with your unique API key
6. Ensure the lead email token for verification has been added to the endpoint URL
7. Click "Save" to save your new Marketo Webhook

8. Proceed to "Creating Response Mappings", using the response attributes and Marketo field values outlined in the table below:

Response Attribute	Marketo Field
details	emailVerifyDetails
info	emailVerifyInfo
status	emailVerifyStatus

Create Marketo Triggers to Match Business Need

After the Verify webhook has been configured in Marketo, create a *Smart Campaign* containing a *Flow Step* to call the Verify webhook based on the desired trigger(s). Design additional Smart Lists, Smart Campaigns, and Flow Steps which trigger based on the Verify Response Code result to reach desired outcome(s). View the sample Marketo Program designed to unsubscribe leads who have Invalid response codes below. **For support designing a Marketo Program to meet your business need, please contact us at products@etumos.com**

Verify Response Codes
Response codes have four separate types: Valid, Suspect, Invalid, and Indeterminate. Within each type, unique response codes provide detailed information about the status of an email.

Status	Info	Retry	Description
Valid (2xx)			
200	OK - Valid Address	no	The mail address is valid.
207	OK - Catch-All Active	no	The mail server for this domain accepts the address, but it also implements a catch-all policy. For this reason, it is not

			possible to determine if a mail account with this name actually exists, without sending a message and waiting for a reply.
215	OK - Catch-All Test Delayed	yes	The mail server for this domain accepts the address, the Catch-All test returned a temporary error.
Suspect (3xx)			
302	Local Address	no	The mail address lacks the domain qualifier.It may work locally within some organization, but otherwise it is unusable.
303	IP Address Literal	no	The mail address is syntactically correct, but the domain part defines an IP address. This kind of address may work, but is usually only used by spammers, or for testing purposes.
305	Disposable Address	no	The mail address is provided by a disposable email address service. Disposable addresses only work for a limited amount of time, or for a limited amount of messages.
308	Role Address	no	The mail address is a role address and typically not associated with a particular person.
313	Server Unavailable	yes	The mail server for this domain could not be contacted, or did not respond.
314	Address Unavailable	yes	The mail server for this domain responded with an error condition for this address.
317	Server Reject	no	The server refuses to answer to SMTP commands, probably because some very strict anti-spam measures are in effect.
Invalid (4xx)			
401	Bad Address	no	The mail address failed to pass basic syntax checks.
404	Domain Not Fully Qualified	no	The mail address is syntactically correct, but the domain part of the mail address is not fully qualified, and the address is not usable.
406	MX Lookup Error	no	There is no valid DNS MX record associated with this domain, or one or more MX entries lack an A record. Messages to this domain cannot be delivered.

409	No-Reply Address	no	The mail address appears to be a no-reply address, and is not usable as a recipient of email messages.
410	Address Rejected	no	The mail server for the recipient domain does not accept messages to this address
413	Server Unavailable	no	The mail server for this domain could not be contacted, or did not accept mail over an extended period of time.
414	Address Unavailable	no	The mail server for this domain responded with an error condition for this address over an extended period of time.
420	Domain Name Misspelled	no	The domain name is probably misspelled.
Indeterminate (1xx)			
114	Validation Delayed	yes	SMTP address validation is still in progress. Retry again after 15 minutes.
118	Rate Limit Exceeded	yes	The API rate limit for your account has been exceeded.
119	API Key Invalid or Depleted	no	The API key is invalid, or the account balance is depleted.

Sample Marketo Program

The Marketo program pictured below is designed to unsubscribe leads with invalid response codes. Please contact Etumos at products@etumos.com for support configuring a Marketo program to meet your business needs.
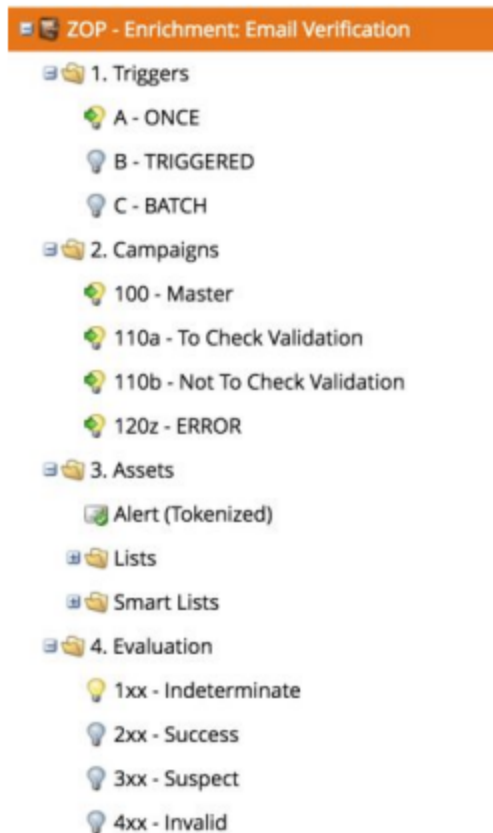
Test Marketo Verify Implementation

To test that the Verify webhook has been configured correctly, create a simple Smart Campaign with a Flowstep which triggers Verify to validate a test lead email address. It's recommended to repeat this flow for an invalid email address, and a valid email address for additional confirmation.

For additional assistance testing your Marketo Verify implementation please contact products@etumos.com.

Verify Troubleshooting & Support

Review the troubleshooting scenarios below to resolve common problems. For additional assistance please contact products@etumos.com

- Problem: The Verify script has been added to my landing pages with Marketo Forms 2.0, but it is not proactively rejecting the email domains I have specified. Common reasons for this problem include:
  - The IP of the page where Verify Javascript snippet is running has not been whitelisted by Etumos.
  - The Verify script url does not contain an active, authorized API key
  - The Verify script is missing essential code to properly run on the page where Marketo Forms 2.0 exists.

- Solutions:
  - Confirm the IP of the page matches that which was given to Etumos to whitelist
  - Confirm subscription to Etumos Verify is active and has not lapsed
  - Review Javascript snippet to ensure all essential code exists and tags are used properly