

## Vector Magic API v1.2

### Revision History

- 2/26/08: Document created
- 2/17/09: Updated to “Developer” instead of “Licensee” to better reflect the actual relationship
- 3/18/09: Added header/footer. Changed version number to 1.0.
- 4/10/09: Added auto-detection of complexity. Added comment about transparency not currently handled by the online tracing engine.
- 3/16/10: Updated document to reflect changes from transition to cloud-based hosting for vectormagic.com. Changed version number to 1.1.
- 10/18/2010: Added EPSZ, SVGZ, and PDFZ read formats. All API users are strongly encouraged to switch to these new gzipped formats. Version 1.2.

### Introduction

The Vector Magic API (“API”) is an online CRUD/REST-style web-API for integration by a third party (“Developer”) with the Vector Magic auto-tracing services (“VM”). The design goals are that the API should:

1. Be simple.
2. Be secure.
3. Provide a smooth user experience (UX).

The subsequent sections describe the theory of operation as well as the specific API call reference documentation.

All API calls consist of standard HTTP requests (GET, POST) to specific URLs together with certain parameters. All responses are either JSON encoded strings (embodying a hash with key-value pairs), or a file (e.g. the vectorized result).

JSON (JavaScript Object Notation) is a very simple and compact encoding for simple data types. It is supported in virtually all languages. Please see <http://www.json.org/> for an introduction and support libraries.

*JSON was chosen instead of XML as it's generally easier to generate and work with and has less overhead.*

## Basics

### Signing up

When signing up for a developer account on the VM website you are issued an id and key. These should be kept confidential as anybody who has them can make calls to the VM API and consequently cause charges to be incurred by Developer.

### Security

The API contains several security features. These are discussed in depth later in the document, while this section covers the information necessary for the integration implementation. The security features come as parameters to the method calls and are common for all API functions. These are:

Parameter	Range / Format / Typical / Default	Description
licensee_id	Integer 1	The ID issued during the signup process.
sequence_number	Integer 1	A Developer-picked sequence number for this request. <i>A negative sequence_number will return an error code when using an unapproved account. This is to allow testing of your error handling.</i> <i>Update as of v1.1: duplicate request checks are no longer done; the sequence number was used for that. It is retained for backwards compatibility.</i>
timestamp	Datetime Wkday, DD MMM CCYY HH:MM:SS GMT Wed, 27 Feb 2008 00:54:45 GMT	RFC2616 date format, see <a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html">http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html</a> . <i>Calls with a timestamp outside of +/- 1 hour of the actual call time will be rejected.</i>
signature	String, 29 characters long. Example: 8AcKoNo8u0N47lrcbL3w8crRj+Q=\n	Base64-encoded RFC 2104 HMAC-SHA1 of the concatenated URL, regular parameters, and security parameters, using the KEY issued during the signup process. See below for examples. See <a href="http://www.ietf.org/rfc/rfc2104.txt">http://www.ietf.org/rfc/rfc2104.txt</a> for the definition of HMAC. <i>Calls with an invalid signature will be rejected.</i>

## Signature for Function Calls

The signature is formed by concatenating the URL called, regular parameters, security parameters, and key in sequence. Say that you are calling the url "http://host.com/path/foo" with the regular parameters a = "bar" and b = "gazonk", then you'd form the string to checksum like this (in pseudo-code):

```
string_to_sign = "http://host.com/path/foo" + "bar" + "gazonk" +  
licensee_id.toString() + sequence_number.toString() + timestamp;  
  
signature = Base64.encode64(HMAC::SHA1.digest(licensee_key,  
string_to_sign));
```

The security parameters listed above are always the LAST parameters in any call signature and shall always be in the order given for the purposes of signing. Don't insert any additional spaces and integers should not be zero-padded (i.e. 1 => "1").

## Signing GET requests

Some of the functions accept HTTP GET requests. The parameters are normally encoded in a *query string* appended to the base URL (i.e. <http://host.com/path/foo> -> <http://host.com/path/foo?a=bar&b=gazonk>). For the purposes of signing the function calls in this case, use the plain base URL and the parameters just like when signing POST requests, do not modify the URL to include the parameters for the purposes of signing.

## Security Discussion

Please see the end of this document for further information regarding security issues.

## Sample code

A sample implementation is provided in Ruby on Rails (see <http://vectormagic.com/developer> for info on how to get the sample code). Samples in other programming languages may be provided in the future.

## Modes of Use

There are several different ways in which a CRUD/REST-style API can be used. With the currently implemented feature set, there is essentially only one mode of use that makes sense: automated job submission. This mode of use is described in detail in the sections below. More modes of use are expected to be supported in future versions of the API.

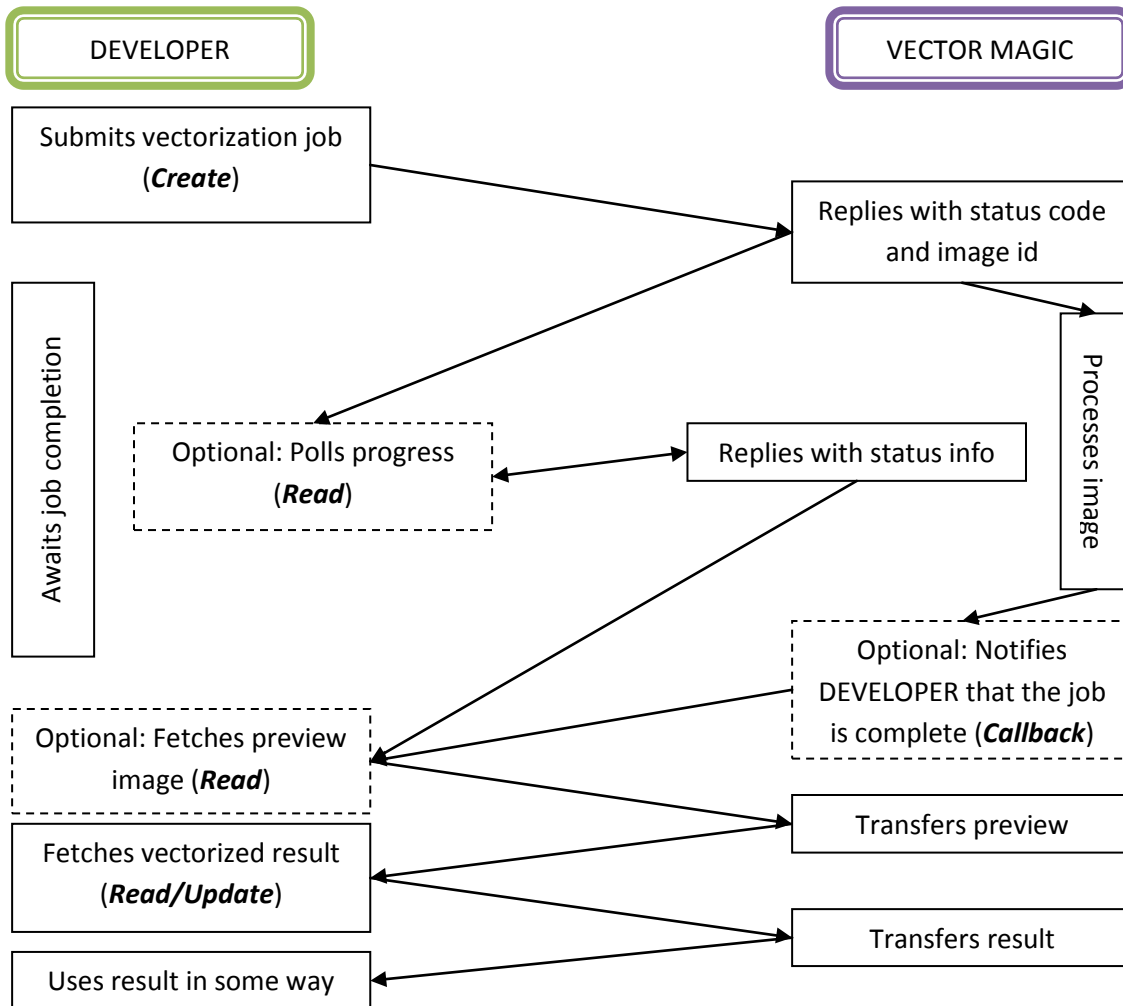
## Automated Job Submission

Developer can submit individual images for tracing by VM, without involving a VM-provided user interface. This mode of use is suitable for Developers who:

- Provide their own user interface to their users, OR
- Somehow knows the suitable parameters to use when vectorizing, OR
- Have found the parameter auto-detection to be reliable enough for production use

## Theory of Operation

The control flow is outlined below. *The function calls corresponding to the different Developer actions are indicated in bold italics.*



## Call Signatures

The functions exposed by the API are:

- **Create** – allows you to submit a new image, and specify the settings to use when vectorizing it.
- **Read** – allows you to fetch state information about an image in various formats: JSON for progress information, PNG for the preview, EPS/SVG/PDF for the vectorized result.
- **Update** – allows you to update some of the state information associated with an image. Currently only used to change the `expire_at` value for an image (allows you to mark it for deletion).

- **Callback** – an optional Developer-provided callback URL where notifications of certain events (e.g. job completion) will be posted.

## Create

This function allows you to submit images for processing.

**URL:** <http://vectormagic.com/api/create>

**Request type:** multipart HTTP POST

Parameter	Range / Format / Typical / Default	Description
image	File	The image to be vectorized.  <i>The image itself is not included when computing the signature. Instead the separate image_checksum parameter is used.</i>
image_checksum	String, 32 characters long 098f6bcd4621d373cade4e832627b4f6	The MD5 checksum of the image file contents.  <i>This parameter is used to allow staged computation of the signature. First compute the file's checksum, and then compute the overall signature.</i>
start_job  Must be set to "vectorize" for the current version	String vectorize	The job to perform on the image.  <i>This parameter is included for forward-compatibility. Future versions of the API will allow involving an interface for the end user, in which case a create operation will need to be issued w/o necessarily starting a job.</i>
image_type  <i>Optional</i>	String <u>auto</u>   photo   logo_aa   logo	The image type to use for the conversion. The default is "auto" which means that it will be automatically detected.
image_complexity  <i>Optional</i>	String <u>auto</u>   high   medium   low	The quality/complexity setting to use. The default is "auto" which means that it will be automatically detected.
image_num_colors  <i>Optional</i>	String   Integer <u>auto</u>   many   2..12	The number of colors to use when vectorizing the image. The default is "auto" which means that the colors are assumed to be "few" (2-12 colors, "Custom" in the wizard) and their number will be automatically detected.

		<p>“Many” maps to “Unlimited” in the wizard.</p> <p>2..12 allows you to fix the number of colors</p>
<p>image_colors</p> <p><i>Optional</i></p>	<p>String</p> <p><u>auto</u>  [comma-separated list of AARRGGBB colors]</p> <p>FF000000 FFFFFFFF FFCC0033</p>	<p>The actual colors to use when vectorizing the image.</p> <p>Ignored when image_num_colors is “auto” or “many”.</p> <p>The default of “auto” means that they will be automatically estimated.</p> <p>AARRGGBB = Alpha, Red, Green, Blue channels in fixed-format two byte hex (00-FF) each. An alpha of FF is fully opaque; an alpha of 00 is fully transparent. If your image doesn’t have an alpha channel, supply alphas of FF.</p> <p><i>Transparency is not currently supported by the tracing engine. Alpha values are required for forward-compatibility reasons.</i></p>
<p>expire_at</p> <p><i>Optional</i></p>	<p>Datetime</p> <p>Wkday, DD MMM CCYY HH:MM:SS GMT</p> <p>Wed, 27 Feb 2008 00:54:45 GMT</p> <p>Default: 2 weeks from now</p>	<p>RFC2616 date format, see <a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html">http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html</a>.</p> <p>The time when the image will expire. VM may delete the image after this time. Any storage charges will count up until this time.</p> <p>Values in the past will be capped to the present.</p>

**Response:** A JSON encoded dictionary with the following key-value pairs:

Parameter	Range / Format / Typical / Default	Description
status	String ok error	Indication of if the call succeeded (“ok”) or failed (“error”).
error_code	Integer	See the error codes section below.
error_message	String	Human-readable error message.
<i>Only included if the call failed</i>		
<i>Only included if the call failed</i>		

image_id	Integer	The id for the image. Used in subsequent show/preview/result calls.
progress	Integer <0 0 1..99 100	Progress indication. <0 => failure (see below) 0 => job not yet started (in queue) 1-99 => job in progress 100 => job completed  <i>The number roughly matches the percent complete.</i>
expire_at	Datetime Wkday, DD MMM CCYY HH:MM:SS GMT Wed, 27 Feb 2008 00:54:45 GMT	RFC2616 date format, see <a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html">http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html</a> .  The time when the image will expire. VM may delete the image after this time. Any storage charges will count up until this time.

Progress failure code	Meaning
-1	Job was cancelled <i>There is currently no cancelling mechanism for the web API. This status code is reserved for such future use.</i>
-2, -3	The conversion failed <i>This type of failure is deterministic (i.e. the same input will always fail), so the job SHOULD NOT be resubmitted without changing the parameters.</i>
-4	Cluster failure <i>Contact support if this happens.</i>
-5	No job associated with this image. <i>This is not necessarily an error condition per se, unless a job has specifically been requested to be started.</i>

## Read

This function allows you to fetch state information about an image in various formats: JSON for progress information, PNG for the preview, EPS/SVG/PDF for the vectorized result.

**URL:** <http://vectormagic.com/api/read>

**Request type:** HTTP GET (POST also accepted)

Parameter	Range / Format / Typical / Default	Description
image_id	Integer 1	The id of the image returned in the "Create" call.
format	String <u>J</u> SON PNG EPS SVG PDF EPSZ SVGZ P DFZ	The format to return the result in. See below.

**Response:** JSON => The same response as in the Create call, PNG => the preview image or a JSON error message (see below), EPS|SVG|PDF => the vectorized result or a JSON error message (see below), EPSZ|SVGZ|PDFZ => the vectorized result in gzipped format or a JSON error message (see below).

*The gzipped formats (EPSZ, SVGZ, PDFZ) are strongly preferred as they save on server load and bandwidth required to transmit the results. The unzipped versions are maintained for backwards compatibility only and new API users are kindly requested to only use the gzipped versions.*

If the preview or vectorized result cannot be fetched (e.g. in case the job failed, has not completed or the image has been deleted) an HTTP error code (404) is returned together with a JSON encoded dictionary with the following key-value pairs:

Parameter	Range / Format / Typical / Default	Description
status	String ok error	Indication of if the call succeeded ("ok") or failed ("error").
error_code	Integer	See the error codes section below.
error_message	String	Human-readable error message.

## Update

URL: <http://vectormagic.com/api/update>

**Request type:** HTTP POST

Parameter	Range / Format / Typical / Default	Description
image_id	Integer 1	The id of the image returned in the "Create" call.
format	String JSON PNG EPS SVG PDF	The format to return the result in. See below.
expire_at <i>Optional</i>	Datetime Wkday, DD MMM CCYY HH:MM:SS GMT Wed, 27 Feb 2008 00:54:45 GMT Default: no change from current value	RFC2616 date format, see <a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html">http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html</a> .  The time when the image will expire. VM may delete the image after this time. Any storage charges will count up until this time.  Values in the past will be capped to the present.

*More parameters may be added in the future. Let us know if there is some specific state information that you want to be able to update for your images.*

**Response:** The same response as in the Read call.

### Additional Comments

- Normally when performing an Update operation on a CRUD/REST-style interface you only get either an ok or an error response back. However, in order to allow Developer to mark an image for deletion and fetch it in the same operation, we provide the same response in the Update operation as in the Read operation. It is not recommended to set the expire\_at timestamp to a point in the past, since it is theoretically possible that a cleanup operation is running at precisely the same time as Developer tries to fetch an image, possibly causing the actual deletion to occur prior to the delivery of the result (exceedingly rare, but could happen). Instead, we recommend setting the expire\_at timestamp to a few minutes out in the future.

### Callback

The callback URL is an optional feature – Developer can register a callback URL with Vector Magic when signing up for the service. This URL will receive notification when vectorization jobs complete. In the future, this may be expanded to provide notification of other TBD events.

You can update the callback URL at any time by updating your Developer account information on the Vector Magic website.

Note that this call goes *from* Vector Magic to Developer.

**Request type:** simple (no multipart) HTTP POST

Parameter	Range / Format / Typical / Default	Description
image_id	Integer 1	The id of the image this notification pertains to.
event	String job_complete	The type of callback. Currently only “job_complete” is supported – Developer should ignore other types of notification for forward compatibility reasons.
status <i>Optional</i>	String ok error	Indication of if the event this notification pertains to succeeded (“ok”) or failed (“error”).
error_code <i>Optional</i>	Integer	See the error codes section below.
error_message <i>Optional</i>	String	Human-readable error message.

**Response:** *The response to the “job\_complete” event will be ignored – future event types may require Developer responses. To not cause forward-compatibility issues, be sure to respond with an empty response body.*

### Additional Comments

- Vector Magic will sign the call in the same way Developer signs its calls to Vector Magic. This allows Developer to reject illicit calls by third parties.
- The “job\_complete” event type will always have a status code associated with it, but future event types may not.
- Your code should not assume that the status, error\_code or error\_message parameters will always be present.
- Your code should ignore unknown event types to allow for expanding the set of notifications without breaking existing code.

### Error Codes

The error codes that may be returned when making calls to the Vector Magic API are listed in the table below. To simplify the parsing, they are grouped into loosely coupled categories/series.

Code	Error	Meaning
<b>40xx: API &amp; security basics</b>		
4000	Bad method	The HTTP method used in making the call was not a POST (it was e.g. a GET)
4001	Timestamp format	The format of the timestamp security parameter did not follow RFC 2616.
4002	Timestamp value	The timestamp is outside of the +/- 1 hour range. Make sure the calling server has its clock set accurately (consider using NTP if you're not already doing so).
4003	Duplicate request	<i>No longer returned</i>
4004	Invalid parameter	An unexpected parameter was supplied in the call. For security reasons, only the parameters listed with each function are allowed.
4005	Missing parameter	A required parameter was missing.
4006	Bad signature	The signature used in the call did not match the one computed by the API server.
<b>41xx: Checks of regular parameters</b>		
4100	Start job value	The “start job” parameter was not acceptable.
4101	Invalid image	The image supplied in the call was either missing or invalid.
4102	Bad image checksum	The checksum of the image file supplied in the call did not match the checksum computed by the API server.
4103	Image type value	The supplied “image type” parameter was not acceptable.
4104	Image complexity value	The supplied “image complexity” parameter was not acceptable.
4105	Image num colors value	The supplied “image num colors” parameter was not acceptable.
4106	Image colors format	The format of the “image colors” parameter was not acceptable.
4107	Image colors value	The number of colors provided in the “image colors” parameter did not match the “image num colors” parameter.
4108	Expire at format	The “expire at” parameter did not conform to RFC 2616.

<b>42xx: Requesting missing or expired data</b>		
4200	Developer not found	The ID provided was not found.
4201	Image not found	The requested image was not found.
4202	Job not found	The requested job was not found.
<b>43xx: Developer approval and quotas</b>		
4300	Not approved	<i>No longer used as of v1.1. The uploaded image will be replaced by a test image for accounts that have not been approved.</i>
4301	Max API calls limit exceeded	<i>No longer used as of v1.1.</i>
4302	Max concurrent jobs limit exceeded	Developer has too many jobs in progress to submit another request. Email to have your quotas raised.
4303	Max submitted images limit exceeded	<i>No longer used as of v1.1.</i>
4304	Max fetched results limit exceeded	<i>No longer used as of v1.1.</i>
<b>5xxx: internal server issues</b>		
5000	Internal server error	An internal server error has occurred. This should normally never happen – contact us immediately if it does.
5001	Cluster overloaded	The vectorization cluster is currently overloaded and cannot accept new jobs at this time. Please try again in a few minutes. If you keep getting this error code, contact us.

## Testing Error Handling

If a negative sequence\_number is supplied, an error code of abs(sequence\_number) will be returned for accounts that have not yet been approved.

Using this feature you can simulate different error conditions and test how your code handles them.

## Security Discussion

Since there will be money involved it is crucial that the authenticity of all calls can be checked and that there are no known security holes. There are several types of attacks that can be envisioned and need to be guarded against:

1. An attacker wishes to vectorize images at Developer's expense.
2. An attacker wishes to incur additional expense for Developer by issuing frivolous calls to Vector Magic.
3. *Any other TBD form of attack that can be envisioned.*

## Guarding against an Attacker Vectorizing images at Developer's Expense

This is guarded against by Developer signing the vectorization call. Since only the Developer and Vector Magic have the key, only they can sign the server side calls. Think of the signature as a per-call password

– an attacker getting one or more of these will not help him generate any new requests of his own choosing.

## Expiration of Old Images

Vector Magic will store the images and associated settings and results supplied by Developer until the `expire_at` time. This is the guaranteed time period - it may be available for much longer.

If you need the result for an image that may have expired we recommend first trying to fetch the result. If you get an error code you can re-submit the image.

**Important:** when re-issuing a job the result may come back different if we have made modifications to the tracing engine in the meantime. Make sure to manage your `expire_at` timestamps to avoid this situation (users will probably end up disappointed if there is a discrepancy between what they've been shown and what is delivered in the end, regardless of if the delivered result is "better").